12-16-2021

Robotic Olfactory-Based Navigation with Mobile Robots

Lingxiao Wang
lingxiaw@my.erau.edu

Robotic Olfactory-Based Navigation with Mobile Robots

by

Lingxiao Wang

A dissertation submitted to the Faculty of Embry-Riddle Aeronautical University in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering and Computer Science

Embry-Riddle Aeronautical University
Daytona Beach, Florida
December 2021

Robotic Olfactory-Based Navigation with Mobile Robots

by Lingxiao Wang

This dissertation was prepared under the direction of the candidate's Dissertation Committee Chair, Dr. Shuo Pang, and has been approved by the members of the dissertation committee. It was submitted to the College of Engineering and accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Electrical Engineering and Computer Science.

Shuo Pang, Ph.D.
Committee Chair

Brian Butka, Ph.D.
Committee Member

Hever Moncayo, Ph.D.
Committee Member

Richard Prazenica, Ph.D.
Committee Member

Tianyu Yang, Ph.D.
Committee Member

Radu Babiceanu, Ph.D.
Interim Chair, Electrical Engineering and Computer Science

12/02/2021
Date

James Gregory, Ph.D.
Dean, College of Engineering

12/2/2021
Date

Lon Moller, J.D.
Senior Vice President for Academic Affairs and Provost

12-3-21
Date

Acknowledgements

I want to thank my advisor, Dr. Shuo Pang, for advising me during my Ph.D. and master's training. Dr. Pang always gives me valuable advice in research, paper and proposal writing, and career development. Without his guidance, the finishing of this work would have been much more difficult.

I also want to thank all my committee members for years of guidance and assistance in my Ph.D. life. I want to thank Dr. Brian Butka for his excellent teaching in the robotics class, from which I decided to pursue in robotics fields. Great thanks to Dr. Tianyu Yang for his advice on my career development and his support for my future career. I also want to thank Dr. Hever Moncayo for providing an advanced robotic testing lab to verify my algorithms. I appreciate Dr. Richard Prazenica for his valuable inputs in building the robotic platform. In addition, I want to thank Dr. Jinlong Li for suggestions in constructing the source mapping algorithm. Great appreciation for Mr. Ziyu Yin and Mr. Christoph Aoun, who sacrificed their summer vacation to help me in on-vehicle tests.

Finally, I want to thank my parents, Yongjun Wang and Jinyan Li, for their selfless support and unconditional love. They are always by my side when I face difficulties and help me come through challenges. I want to especially thank my grandmother, Lizhi Liu, who cared for me since my childhood. My grandmother did not receive high education, but she knew the importance of education. She taught me to become a caring, friendly, and loving person. I want to dedicate this work to my family for their love and support.

Contents

List of Figures

List of Tables

Abstract

Robotic odor source localization (OSL) is a technology that enables mobile robots or autonomous vehicles to find an odor source in unknown environments. It has been viewed as challenging due to the turbulent nature of airflows and the resulting odor plume characteristics. The key to correctly finding an odor source is designing an effective olfactory-based navigation algorithm, which guides the robot to detect emitted odor plumes as cues in finding the source. This dissertation proposes three kinds of olfactory-based navigation methods to improve search efficiency while maintaining a low computational cost, incorporating different machine learning and artificial intelligence methods.

A. Adaptive Bio-inspired Navigation via Fuzzy Inference Systems.

In nature, animals use olfaction to perform many life-essential activities, such as homing, foraging, mate-seeking, and evading predators. Inspired by the mate-seeking behaviors of male moths, this method presents a behavior-based navigation algorithm for using on a mobile robot to locate an odor source. Unlike traditional bio-inspired methods, which use fixed parameters to formulate robot search trajectories, a fuzzy inference system is designed to perceive the environment and adjust trajectory parameters based on the current search situation. The robot can automatically adapt the scale of search trajectories to fit environmental changes and balance the exploration and exploitation of the search.

B. Olfactory-based Navigation via Model-based Reinforcement Learning Methods.

This method analogizes the odor source localization as a reinforcement learning problem. During the odor plume tracing process, the belief state in a partially observable Markov decision process model is adapted to generate a source probability map that estimates possible odor source locations. A hidden Markov model is employed to produce a plume distribution map that premises plume propagation areas. Both source and plume estimates are fed to the robot. A decision-making model based on a fuzzy inference system is designed to dynamically fuse information from two maps and balance the exploitation and exploration of the search. After assigning the fused information to reward functions, a value iteration-based path planning algorithm solves the optimal action policy.

C. Robotic Odor Source Localization via Deep Learning-based Methods.

This method investigates the viability of implementing deep learning algorithms to solve the odor source localization problem. The primary objective is to obtain a deep learning model that guides a mobile robot to find an odor source without explicating search

strategies. To achieve this goal, two kinds of deep learning models, including adaptive neuro-fuzzy inference system (ANFIS) and deep neural networks (DNNs), are employed to generate the olfactory-based navigation strategies. Multiple training data sets are acquired by applying two traditional methods in both simulation and on-vehicle tests to train deep learning models. After the supervised training, the deep learning models are verified with unseen search situations in simulation and real-world environments.

All proposed algorithms are implemented in simulation and on-vehicle tests to verify their effectiveness. Compared to traditional methods, experiment results show that the proposed algorithms outperform them in terms of the success rate and average search time. Finally, the future research directions are presented at the end of the dissertation.

Chapter 1

Introduction

1.1 Problem Statement

Humans perceive the world with five senses, including sight, sound, smell, taste, and touch. In robotic applications, a machine can achieve similar perception capabilities by integrating various kinds of sensors. Autonomous driving vehicles [1], for instance, can see and hear the surrounding environment via cameras, LIDAR sensors, and sonar. However, compared with other sensing abilities, the sense of smell, i.e., olfaction, has not been thoroughly studied in the robotics literature.

Robotic odor source localization (OSL) is a technology that employs a mobile robot or an autonomous vehicle, equipped with odor detection sensors, i.e., chemical sensors, to find an odor source in an unknown environment [2]. This technology enables robots with a sense of smell, i.e., olfaction, to trace and find odor sources, which has many practical applications. Some of them that are frequently quoted include monitoring air pollution [3], locating chemical gas leaks [4], locating unexploded mines and bombs [5],

finding survivors in search-and-rescue operations [6], and marine surveys such as finding underwater hydrothermal vents [7].

To correctly find an odor source, an effective olfactory-based navigation algorithm is critical. Like image-based navigation algorithms, which utilize the information extracted from images as references to navigate a robot, olfactory-based navigation algorithms detect odor plumes as cues to guide robots in finding the odor source. The challenge of this navigation problem is to command robots to detect odor plumes continuously. In a turbulent airflow environment, the turbulence of the fluid medium constantly stretches and twists the odor filaments inside an odor plume, and the temporal and spatial variations in the airflow velocity cause the plume centerline to meander. The turbulent airflows make the odor concentration distribution far from a smooth gradient along the wind direction, but an intermittent and patchy signal [8]. Therefore, olfactory-based navigation over long distances in real-world turbulent fluid fields is not trivial.

This thesis presents a variety of olfactory-based navigation algorithms for implementation on a mobile robot to find an odor source in unknown environments. The research niche of this work is using modern artificial intelligence (AI) and machine learning (ML) algorithms to improve the search efficiency while retaining the computational simplicity. By using these methods, including fuzzy inference theories, reinforcement learning (RL), adaptive neuro-fuzzy inference systems (ANFIS), and deep neural networks, the plume tracing robot can intelligently adjust its search behaviors and swiftly plan its search routes depending on the current search situation. Experiment results show that the proposed algorithms improve the search time and success rate compared to traditional olfactory-based navigation methods.

1.2 Background and Motivation

It is commonly accepted that an OSL problem can be divided into three phases, namely plume finding, plume tracing, and source declaration [9]. In the first phase, the robot searches for the presence of odor plumes. After the robot detects plumes, it switches to the plume tracing phase, which follows plumes as cues to find the odor source. In the source declaration phase, the robot recognizes the odor source and declares the odor source location.

The key step to solve an OSL problem is the plume tracing phase, where olfactory-based navigation algorithms are designed to guide robots to detect plumes and move to the odor source. The challenging part of this navigation problem is to estimate the emitted odor plume locations, which are not only related to the molecular diffusion that takes plumes away from the odor source but also the advection of airflows [8]. In laminar flow environments, plume dispersal is a steady and stable process, which results in a spatially coherent plume trajectory. In this scenario, the intuitive gradient following algorithm, i.e., chemotaxis [10], is applicable for guiding a plume tracing robot to find the odor source. However, in a turbulent flow environment, odor plumes are stretched and twisted to form an intermittent concentration trajectory, which fails the chemotaxis in this environment. Alternatively, two other categories of olfactory-based navigation algorithms, namely bio-inspired and engineering-based (i.e., probabilistic) algorithms, have been researched and proposed [11].

A bio-inspired algorithm directs the robot to mimic animal olfactory behaviors. A typical bio-inspired algorithm is the moth-inspired method, which is inspired by the mate-seeking behaviors of male moths [12]: a male moth flies against wind when detecting pheromone plumes emitted from female moths and across wind when plumes are absent.

This behavior can be framed as a 'surge/casting' model, where a plume tracing robot moves against the wind direction when detecting plumes (i.e., 'surge') and traverses the wind when missing plume contact (i.e., 'casting') [13]. By contrast, engineering based navigation algorithms utilize mathematical and physics-based approaches to deduce possible odor source locations. The main procedures are twofold [14]: first, possible odor source locations are estimated via source mapping algorithms; then, a path planner is employed to direct the robot to the estimated target.

Compared to the aforementioned algorithms, chemotaxis is barely researched since it is not applicable in turbulent flow environments. Bio-inspired algorithms are simple and easy to implement but not as effective as an engineering-based algorithm in highly turbulent flow environments. When the airflow field varies significantly, the simple 'surge/casting' model can barely keep the robot continuously detecting plumes [15]. Engineering-based algorithms, on the other hand, can estimate odor source or plume locations to facilitate the plume tracing process and improve the search efficiency [16], but the high computational cost to estimate source or plume locations online impedes its applications on robotic platforms, which have limited computational resources. In summary, a desired olfactory-based navigation algorithm should be light-weight, i.e., it does not require high computational resources, while efficient and capable enough to find odor sources in different airflow environments.

1.3 Thesis Overview

This thesis is organized as below: Chapter 2 presents a comprehensive review of recent olfactory-based navigation algorithms; Chapter 3 shows the adaptive bio-inspired

navigation method using fuzzy inference methods; Chapter 4 presents the reinforcement learning-based navigation method; Chapter 5 demonstrates the data driven-based methods, including deep learning and adaptive neuro-fuzzy inference systems; Chapter 6 presents the setup and results of on-vehicle tests; Chapter 7 shows the conclusion of this work and future works.

Chapter 2

Literature Review

Robotic OSL is an emerging research topic in recent decades. Thanks to the rapid development of robotics and autonomous systems, employing mobile robots or autonomous vehicles to trace and locate odor (or chemical) sources in harsh environments becomes feasible. Over the past three decades, a significant amount of research works have proposed various kinds of olfactory-based navigation algorithms. This chapter provides a survey that reviews recently published olfactory-based navigation algorithms, grouped into four categories: chemotaxis, bio-inspired, engineering-based, and multi-agent search algorithms.

2.1 Chemotaxis: Gradient Following Algorithms

Early robotic OSL algorithms attempted to find odor sources via a simple gradient following algorithm, i.e., chemotaxis. In nature, bacteria such as Escherichia coli use the chemotaxis to locate nutrients [17], where they can sense the chemical concentration

and move toward the area with the high concentration. Fig. 2.1 illustrates the principle of chemotaxis algorithms.

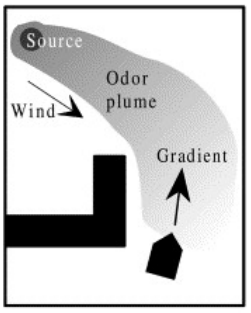


FIGURE 2.1: A chemotaxis method. The gradient of odor concentration is used as the reference to navigate the robot. Reprinted from [18], Fig. 1.

To implement the chemotaxis method, Sandini *et al.* [19] employed a mobile robot equipped with a pair of chemical sensors, and the robot was programmed to steer toward the side with the higher chemical concentration. The chemical concentration needs to be monitored periodically to guarantee that the robot moves toward the source, i.e., the continuous increasing concentration is interpreted as the robot approaching the source, and the consistent falling concentration indicates the robot leaving the source. Many indoor experiments [20–23] have proved the validity of chemotaxis in laminar flow environments (i.e., low Reynolds numbers), where the chemical concentration is a smooth and stable signal. However, this method is not applicable in an environment with turbulent flows (i.e., high Reynolds numbers) since odor plumes are congregated into packets and the gradient of concentration becomes a patchy and intermittent signal [24]. To approach the problem of plume tracing in turbulent flow environments, bio-inspired and engineering-based methods were proposed and researched.

2.2 Bio-inspired Methods

The core idea of bio-inspired methods is commanding a robot to mimic successful odor tracing and localization behaviors of animals. Although gradient-based algorithms (i.e., chemotaxis) are inspired by living creatures, they are mostly related to olfactory behaviors of lower organisms like bacteria. In turbulent flow environments, algorithms inspired by more complicated creatures, e.g., moths, lobsters, and beetles, are more reliable. This section reviews three major bio-inspired methods, including moth-inspired, lobster-inspired, and beetle-inspired methods.

2.2.1 Moth-inspired Methods

2.2.1.1 The 'Surge/Casting' Model

Most bio-inspired methods concentrate on moth-inspired methods (i.e., anemotaxis), which is inspired by the mate-seeking behavior of male moths [25]. Research shows that male moths could successfully locate female moths by tracking pheromone plumes emitted by female moths over a long distance [12, 26] and overcoming obstacles such as forests [27].

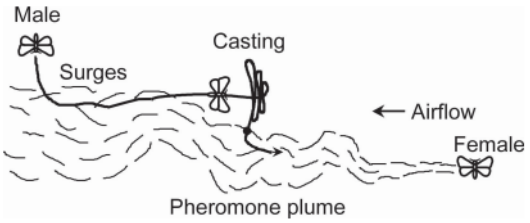


FIGURE 2.2: A male moth traces a female moth through pheromone plumes using the anemotaxis method. Reprinted from [28], Fig. 3.

Fig. 2.2 shows how a male moth finds a female moth via tracing emitted pheromone plumes: when a male moth detects pheromone plumes, it tries to maintain contact with

the plume by moving upwind, termed 'surge'; when the contact of plumes is missed for a sufficiently long period of time, it pauses the upwind movement and performs crosswind excursions, termed 'casting'. These two main steps comprise the fundamental frame of the anemotaxis method (i.e., the 'surge/casting' model).

2.2.1.2 Implementations of Moth-Inspired Methods

Early work on moth-inspired methods verified the feasibility of the 'surge/casting' model. Ryohei *et al.* [29] generalized the 'surge/casting' model by observing moth reactions under pheromone stimulus and implemented it on a wheeled ground vehicle to find an odor source in a closed environment. Yoshihiko and Isao [30] presented a decision and control model based on a recurrent neural network, which commanded the robot to imitate the 'surge/casting' behaviors. In their method, inputs from two gas sensors were processed by the proposed neural network and transmitted to two motors to control the robot. A similar controller based on a more complex neural network with more layers and inputs was proposed in [31]. Simulations showed that the proposed neural network controller was valid in both low-turbulent and high-turbulent environments. Lochmatter *et al.* [32] implemented this behavior pattern on a wheeled ground vehicle to find an odor source in a laminar flow environment.

In the underwater environment, Li *et al.* [33] implemented the 'surge/casting' model on an autonomous underwater vehicle (AUV) to search for an underwater chemical source. In the surge behavior, the AUV was commanded to move up-flow when it was in the plume, and when it left the plume, it steered an offset angle to change its heading and move forward. If the AUV re-contacted the plume, it went back to the surge behavior; otherwise, it moved in a 'cloverleaf'-like trajectory to re-detect plumes in the 'casting' behavior. Results from water experiments [34, 35] showed that the AUV successfully

found the source, which proved the validity of the moth-inspired method in underwater OSL applications. Above the ground, Luo *et al.* [36] designed a flying odor compass based on an unmanned aerial vehicle (UAV), which can identify the airflow direction based on plume detection events and trace plumes to find the odor source accordingly (i.e., fly upwind).

2.2.1.3 Some Modifications

Recent developments of moth-inspired methods tried to modify the conventional 'surge/-casting' model to obtain a better search performance on robotic systems. Farrell *et al.* [37] modified the conventional 'surge' behavior by adding customized 'track-in' and 'track-out' behaviors to increase the plume contact duration in the plume tracing process. When the AUV (employed as the robotic agent in this work) loses the plume detection, it will first search plumes in the surrounding area before turning to the 'casting' behavior, i.e., 'track-out' behavior. Once the AUV finds odor plumes, it turns to the 'track-in' behavior again and moves upflow to the source.

Shigaki *et al.* [38] introduced a time varying moth-inspired method, where the duration of the 'surge' behavior is calculated via an equation obtained by analyzing the moth's muscle activities under odor stimulation. They found that the duration of the 'surge' behavior of a moth is associated with the number of the stimulus (i.e., plume detection numbers). When a moth detects the pheromone plume for the first time, the surge duration is the longest, and the duration decreases if more plumes are detected. Based on this observation, a math model was created to imitate this mechanism, where the robot 'surge' duration decreases when the plume detection number increases. This model was verified by experiments with a mobile robot, and results from indoor experiments showed that this method had an approximate 90% success rate. Recently, they proposed

a fuzzy controller to adaptively control the transition among 'surge', 'stop', and 'casting' behaviors according to airflow changes in the search environment [39]. This method was implemented on a mobile robot to find an odor source in an indoor environment. Experimental results show that the proposed method improves the success rate by 45.7% compared to the normal 'surge/casting' anemotaxis method.

Shunsuke *et al.* [40] found wind directions also affect moth behaviors when they trace pheromones. Biological experiments showed that moths tend to decrease their speed and angular velocity when they face the wind. A modified 'surge/casting' model with a middle step that checks the current wind direction was proposed, and if the detected wind direction is upwind, the robot will decrease its speed.

Liberzon *et al.* [41] modified the traditional moth-inspired methods to find an odor source in turbulent airflow environments. In their method, a timer is added to count the time interval (i.e., t_c) of the first detected plume and the last detected plume. As long as the plumes are encountered at the rate of t_c, the robot performs the 'surge' behavior (i.e., move upwind). If the plumes do not arrive for a time longer than t_c, the robot performs the 'casting' behavior to re-detect plumes. Simulated results showed that the modified method has an average success rate of about 80%.

Ferri *et al.* [42] proposed a 'spiral' search trajectory for directing the robot to re-detect plumes in the 'casting' behavior. Rahbar *et al.* [43] presented a 3-dimensional (3-D) version 'spiral' trajectory and validated it with experiments in a wind tunnel. Besides, many bio-inspired search strategies have been proposed to modify the conventional 'surge/-casting' behavior model, such as the multi-phase exploratory method [44].

2.2.2 Lobster-inspired Methods

Odor search strategies in lobster-inspired methods are similar to the conventional gradient-based algorithms but take inspiration from lobster's foraging strategies. Lobsters have a pair of noses, which are located on their lateral antennules. When a lobster forages for food, it uses antennules to guide its progress and adopts the following two strategies: 1) the lobster turns towards the side of higher odor concentration or goes straight forward if two antennules detect nearly the same concentration; 2) the lobster moves backward if neither of two antennules detects concentration [45].

Lobster-inspired methods are usually implemented on underwater robots (e.g., AUVs) to trace plumes and locate the odor source in underwater environments. Consi *et al.* [46] designed an odor plume following AUV that imitates lobster foraging behaviors. In experiments, the AUV was placed in a fish tank with fresh water and seawater was injected into the fresh water tank to represent odor plumes. The AUV was equipped with a pair of conductivity sensors to detect seawater (i.e., odor plumes). Similar to the lobster's foraging behavior, the AUV moves forward when both sensors detects odor plumes and turns to the side with the higher sensed concentration. Experimental results demonstrated the validity of lobster-inspired methods in following odor plumes to the source.

Compared to the conventional chemotaxis method, the use of two spatially separated sensors and the addition of a second strategy confer a significant advantage. Grasso *et al.* [47] implemented the lobster-inspired method on an AUV in a turbulent underwater environment. They compared the conventional chemotaxis and the lobster-inspired method through experiments and found that the second strategy in the lobster-inspired method (i.e., the lobster moves backward if neither sensors detects concentration) was

necessary to keep the AUV in the plume long enough, enabling it to trace back to the source. If the AUV was guided solely by left-right concentration differences (i.e., chemotaxis), it was prone to drift out of the plume and unable to re-detect odor concentration. They found that the addition of the second strategy increased the success rate of finding the source from 33% to 66%.

2.2.3 Beetle-inspired Methods

The dung beetle is reported to use a very simple strategy to follow the odor plume emitted from a cowpat: when the beetle detects odors, it zigzags diagonally across the plume in an upwind direction, turning back each time it leaves the edge of the plume. When the odor plume ends, as it does immediately above the cowpat, the beetle flies down and usually lands on or near its goal [21].

Inspired by dung beetles' odor search behaviors, the beetle-inspired methods can be summarized as following: when the robot detects odors, it zigzags diagonally across the plume in an upwind direction, turning back each time it leaves the edge of the plume. Ishida *et al.* [10] applied the beetle-inspired methods to guide a mobile robot in tracing a chemical source. The robot was equipped with tin oxide gas sensors and anemometer sensors to measure the chemical gradient and wind direction.

Lochmatter and Martinoli [48] implemented the 'zig-zag' search pattern on a mobile robot to find an odor source in a wind tunnel. Experiment results revealed that the beetle-inspired method could reach around 90% success rate in 20 test runs, indicating the effectiveness of this method in finding an odor source in a small-scale environment (search area is around 14×4 m^2).

2.2.4 Summary of Bio-inspired Methods

Fig 2.3 summarizes and compares search trajectories of the aforementioned bio-inspired algorithms. In summary, the core of moth-inspired methods is a 'surge/casting' behavior pattern: the robot is commanded to move upwind if odor concentrations are detected and traverse the wind when plumes are absent; lobster-inspired methods are the modified version of chemotaxis, which command the robot to follow the gradient ascent direction and move backward if no concentrations are detected; beetle-inspired methods trace odor plumes via a simple 'zig-zag' search trajectory, where the plume tracing robot moves upwind if plumes are detected.

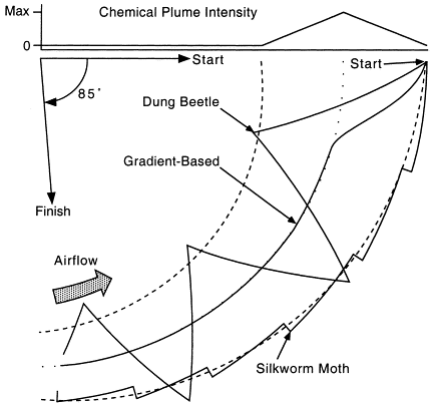


FIGURE 2.3: Search trajectories of bio-inspired odor search methods, including moth-inspired, lobster-inspired (i.e., gradient following), and beetle-inspired (i.e., 'zig-zag') methods. The coordinate on the top of the diagram indicates the relationship between the horizontal distance toward the odor source (x axis) and the odor concentration (y axis). This diagram is reprinted from [21], Fig. 5.

The prime benefit of bio-inspired methods is the simplicity. This type of method does not require learning procedures, complex decision-making, or heavy calculation, making this type of algorithms easy to implement on mobile robots. However, it should also be mentioned that most of the published bio-inspired methods were validated in computer simulations or indoor experiments, where airflow environments are stable and laminar. The reported success rate of these tests is high, and this good algorithm performance

is anticipated since experimental settings were constrained and modified in these experiments. For instance, the exploration area was limited to a small size; electrical fans were used to control the wind speed and the direction; the robot's starting position was located downwind of the gas source. These modifications make it easy for a robot to detect chemical plumes and find an odor source. For outdoor experiments, where the airflow environment becomes turbulent and varying, the simple bio-inspired methods usually do not perform as well as in indoor experiments. This is because animals are fundamentally different from autonomous vehicles in terms of sensing and actuation capabilities [49], which makes a pure bio-inspired method less practical and effective for an odor source localization task in an outdoor environment.

Another limitation of bio-inspired methods is the difficulty of developing them into multi-robotic algorithms since it is hard for robots to collaborate while performing bio-inspired search strategies. For instance, [50] implemented and compared multiple different bio-inspired algorithms, including a beetle-inspired and two moth-inspired methods, to search for an odor source with both single-robot and multi-robot systems. The authors discovered that robots in the multi-robot systems are likely to hinder each other along the way to the source, resulting in a worse search performance than the single-robot counterpart. This is because the uncoordinated robots have trouble overtaking others along their way to the source, e.g., when two robots get close, the obstacle avoidance mechanism is likely to drive one or both of them out of the plume.

2.3 Engineering-based Methods

By contrast with bio-inspired methods, an engineering-based method (i.e., a probabilistic method) does not follow a fixed behavior pattern. It utilizes math and physics

approaches to model odor plume distribution and estimates possible odor source locations. A common way to indicate the odor source distribution is constructing a source probability map. This map divides the search area into multiple small regions and assigns every region with a probability value, indicating how likely this region contains the odor source. The odor source estimate is the region with the highest probability. After the source estimate is obtained, the path planner calculates a collision-free search trajectory to guide the robot moving toward the estimated source location. During the robot maneuver, the source probability map is iteratively updated based on a series of odor detection and non-detection events. This section reviews recent published engineering-based methods.

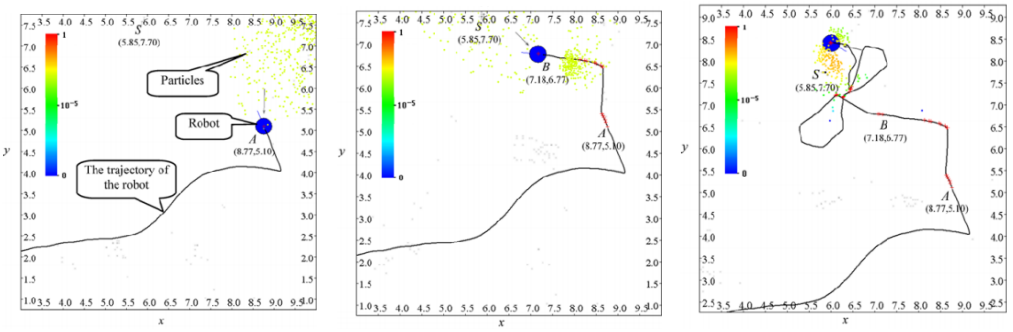
2.3.1 Bayesian-inference Method

Pang and Farrell [51] proposed a recursive algorithm based on the Bayesian method to construct and update the source probability map. The plume propagation was modeled as a Gaussian process, containing a determinate portion (induced by advection) and a random portion (induced by fluctuation). By integrating over the plume propagation time and the region area, the probability of a region containing the odor source is calculated based on the Gaussian plume model and the sensed airflow information. The quality of the source probability map was loop updated based on odor detection and non-detection events. After the source probability map was converged, the region with the highest probability is the final odor source location.

2.3.2 Particle Filter-based Method

Li *et al.* [52] presented an odor source mapping method based on the particle filter (PF) algorithm. PF is a sequential importance sampling filter, which uses a set of particles to empirically represent the posterior distribution given noisy or partial observations. Fig. 2.4 represents the generated source probability map and robot search routes at different time steps in an OSL trial. In this method, regions in the search area are considered as particles, and each particle is assigned with a weighting factor, representing the probability of the particle containing the odor source. Like the Bayesian-inference method, the source probability map was loop updated based on sensed airflow information and plume detection/non-detection events. The weighted mean of the particles was treated as the most possible location of the odor source. The convergence of the particles is used as a termination condition, and the estimated source location is the position where particles converge. During the whole process, the robot conducts a moth-inspired search strategy to trace plumes until the termination condition is satisfied. It can be seen in Fig. 2.4(c) that the estimated odor source location is close to the actual one, i.e., particles are converged to the actual odor source location.

Chen and Huang [53] proposed a PF-based algorithm for tracking smoke plumes using a mobile robot. They modified the standard PF algorithm by integrating the moth-inspired algorithm. In the particle resampling step, the firefly algorithm is introduced to mimic the 'surge' behavior in moth-inspired algorithms, where each particle is regarded as a firefly. The equation to update fireflies ' positions includes an upwind term, guiding fireflies (i.e., particles) to move upwind. A noteworthy improvement is that this method provides an idea that the particle updating process in the standard PF algorithm could be integrated with other olfactory-based navigation methods, resulting in the optimization



(a) t=69 s (b) t=92 s (c) t=203 s

FIGURE 2.4: Source probability maps generated at different time steps in an OSL trial proposed by Li *et al.*. The source mapping algorithm is based on the particle filter algorithm. Particles are represented by different colors to indicate varying probabilities of an area containing the source, i.e., the warmer the color, the higher the probability. The search route was generated by the moth-inspired method. Reprinted from [52], Fig. 6.

improvement.

2.3.3 Hidden Markov Model-based Method

Farrell *et al.* [16] presented a hidden Markov model (HMM) based source mapping algorithm. The plume propagation in the search area is modeled as a HMM, which contains three components: hidden states, state transition distribution, and observation probability distribution. The search area is divided into $m \times n$ grid cells, then the hidden state is defined as whether a cell contains the detectable odor plume. The state transition distribution, which describes the transportation of odor plumes, is defined based on sensed flow information. The observation distribution is the probability of the robot detecting odor plumes in a cell, which is modeled as a constant probability coefficient $\mu \in (0, 1)$. With this HMM framework, inferring the source distribution is an HMM training problem that tries to obtain the initial states according to observations, i.e., estimate the odor source location. This problem could be solved based on classical HMM solutions, such as the forward and backward algorithms. The estimated odor

source location was reversely calculated based on the plume transmission model, which was determined by the sensed flow information.

2.3.4 Partially Observable Markov Decision Process-based Methods

Feng *et al.* [54] adapted a partially observable Markov decision process (POMDP) model to construct the source probability map. In a POMDP model, states are hidden to the agent, and the agent estimates which state it is currently in via the belief state (i.e., a probability indicating how likely the agent is in a state). In [54], the plume tracing process is modeled as a POMDP model, where hidden states are defined as possible odor source locations; the probability of the robot detecting or not detecting odor plumes is used to define observation probabilities; belief states are adapted to represent the source probability map. By iteratively updating the belief states via odor detection and non-detection events, the source probability map is obtained.

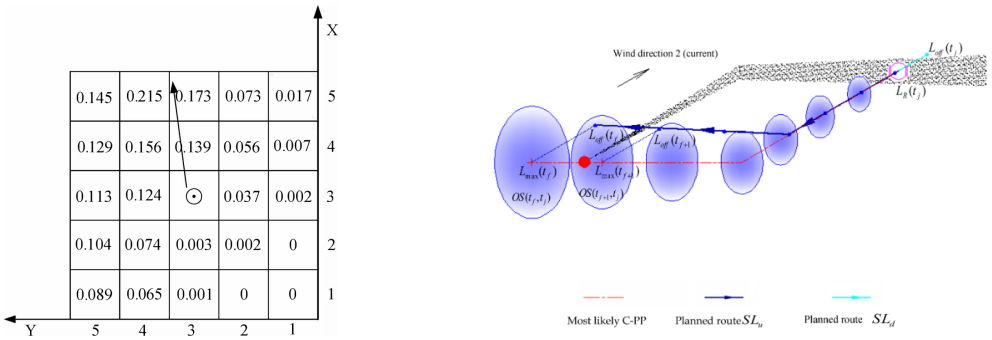
Saigol *et al.* [55] proposed an information-lookahead plume tracing algorithm for usage on AUVs to search multiple underwater odor sources (i.e., hydrothermal vents). The plume tracing process is also modeled as a POMDP, where hidden states are defined as a bundle including AUV position, ocean current vector, and the actual vent locations. Thus, the belief states represent the source probability map, indicating the probability of each region containing an odor source. In the planning procedure, the optimal policy, i.e., a series of optimal AUV actions, is calculated via the information-lookahead algorithm, where the adapted POMDP model is approximately solved by only evaluating actions N steps into the future.

Hu *et al.* [56] presented a model-free reinforcement learning (RL) plume tracing algorithm for usage on AUVs to find an underwater chemical source. Due to the limited

perception and noise in the deep sea environment, the authors argued that an AUV cannot accurately observe environmental parameters, such as AUV positions and flow speeds/directions. Thus, the AUV's plume tracing process is modeled as a POMDP. Instead of solving the adapted POMDP model using traditional methods, the deterministic policy gradient (DPG) algorithm is adopted, where a recurrent neural network (RNN) is employed as the actor and critic networks. Simulation experiments were conducted, and results showed that the proposed method outperforms the dynamic programming method (i.e., model the plume tracing process as an MDP) and the deep deterministic policy gradient (DDPG) algorithm (i.e., use feedforward neural network instead of RNNs).

2.3.5 Infotaxis Method

Vergassola *et al.* [57] presented the 'infotaxis' method, which uses information entropy to guide the robot searching for an odor source. This method contains two core components, namely Bayesian estimation of the source position based on historical detection events and greedy decision making (i.e., path planning) based on entropy minimization. The innovative of this method is that instead of using a binary sensing model, a criterion of odor encounter rate is presented to estimate the posterior probability of odor source based on robot measurements. Given a source location, the mean number of odor encounters at a position is modeled as a random variable subject to a Poisson distribution. In the path planning procedure, Shannon's entropy theory is employed to describe the uncertainty of the odor source location, where the robot is guided to move toward the direction that reduces the information uncertainty most.



(a) (b)

FIGURE 2.5: (a) A path planning algorithm based on the artificial potential field approach proposed by Jiu *et al.* Numbers in cells represent probabilities of cells containing the odor source. The arrow indicates the virtual force, which points the direction from the current robot's location to the estimated source. Reprinted from [**?**]. (b) An online route planning algorithm proposed by Li *at al.* Blue ellipses represent estimated chemical plumes; the gray strip indicates the actual plume trail; the solid blue line represents the planned route. Reprint from [59], Fig. 1.

2.3.6 Path Planners

After obtaining a source likelihood map, a path planning algorithm that guides the robot moving toward the target is necessary. Jiu *et al.* [58] proposed a path planning algorithm based on the artificial potential field (APF), which directs the robot to the region with the highest probability (as shown in Fig. 2.5(a)). The source probability map was obtained via the Bayesian-inference method. Since there were no obstacles in the search area, the local minima issue in the standard APF algorithm was avoided, i.e., when the attractive force equals the repulsive force and with opposite direction, the robot is trapped. From the engineering perspective, Li et al. [59] proposed a route planning method based on the estimated plume trajectories. This method first estimates trajectories of plumes based on a Gaussian gas diffusion model, and then the robot is commanded to move to the area where thick plumes exist.

2.3.7 Summary of Engineering-based Methods

The aforementioned engineering-based methods provide an unique angle to solve the robot OSL in an analytical perspective. The two basic components involved are 1) constructing the source probability map based on the onboard sensor measurements and 2) planning the robot movements based on the source estimate. For the first part, different probabilistic methods (e.g., Bayesian-inference, HMM, PF, POMDP, etc.) and different odor sensing models (e.g., binary measurements, number of encounters, or the concentration) have been introduced to develop more accurate and efficient source mapping algorithms. For the second part, information-driven (e.g., infotaxis) and APF strategies have been employed to make full use of the source mapping results.

However, some limitations of this type of algorithm should also be mentioned. First and foremost, engineering-based methods are always devised along with assumptions, which are usually deviated from reality to some extent. For instance, since the global airflow information is not available, the wind field between the odor source and the plume tracing robot is assumed to be uniform to deduce the plume advection distance in [51]. This assumption may be valid in laminar flow environments but cannot hold when airflow directions vary spatially (this problem could be mitigated by employing multi-agent systems, which is addressed in Section 2.4). If these engineering-based methods are used in OSL applications, one needs to treat these assumptions with caution. Besides, detailed analyses and experiments would prompt these algorithms to develop more reasonable assumptions that fit the corresponding application conditions. In addition, the computational load of engineering-based methods is another concern.

Secondly, engineering-based methods usually require high computational resources to estimate possible odor source locations. Updating the source probability map requires

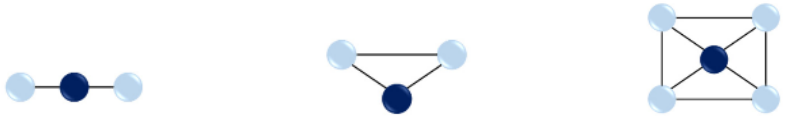
a thorough calculation of the source probability in every region of the search area. When the search area is large and complex, the computational load of engineering-based methods is significant, hindering their implementation on mobile robots, which have limited computational resources.

2.4 Multi-agent Search Algorithms

The multi-agent search algorithm employs more than two mobile robots in an OSL mission. The primary advantage of implementing multiple robots is reduced searching time, which is a vital criterion for OSL applications like search-and-rescue operations. With the increasing number of sensing nodes, environmental information from multiple positions in the search area can be detected simultaneously, improving the search efficiency. In addition, employing multiple-robot systems in an OSL task could also provide a greater robustness against hardware failures. However, the main challenge of this type of algorithm is the appropriate design of an olfaction-based navigation algorithm that coordinates multiple robots to work effectively and efficiently. This section reviews existing multi-agent OSL algorithms, which can be grouped into two categories, including formation-based algorithms and swarm-based algorithms.

2.4.1 Formation-based Algorithms

Inspired by formation behaviors of animals, such as flocking of birds and swarming of ants, formation control algorithms [60] command robots to perform operations collaboratively while maintaining the desired formation. In terms of an OSL task, formation-based OSL algorithms lead a group of robots with a certain spatial distribution to find an odor source while maintaining the formation alignment.



(a) (b) (c)

FIGURE 2.6: Line, triangular, and square formation proposed by Lochmatter *et al.* [61]. The center robot stays on the center line of plumes, while other robots remain at the edge of plumes. The formation is dynamically changed according to the real-time plume shape to ensure the formation is always in the plumes.

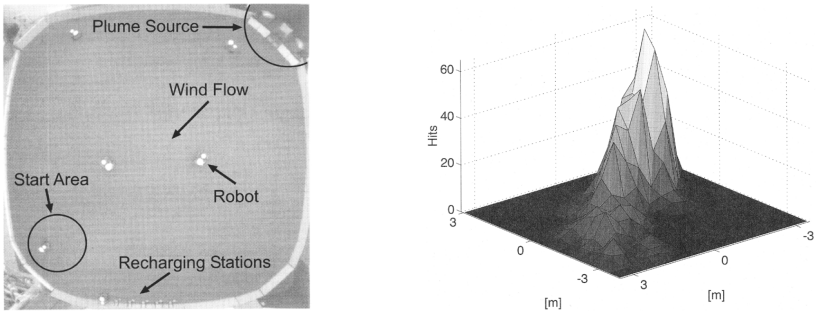
Lochmatter *et al.* [61] proposed various types of formations, including line, triangular and square (as shown in Fig. 2.6). In a formation, the robot in the center will stay on the center line of plumes, while other robots remain at the edge of the plumes. The distance between the robots will change according to the real-time plume shape to ensure that the robot formation is always in the plumes. When robots are in the plume, they move upwind and share their measurements including gas concentrations and wind directions with each other. When robots are out of the plume, they perform the casting search to regain the plume contact. Simulation results revealed that the proposed formations have evinced the capability of successfully tracing plumes and finding the odor source, and the rectangular formation completed the mission in a shorter period than the line formation.

Jorge *et al.* [62] evaluated line and rectangular formations in simulations, where multiple ground mobile robots are controlled via a Laplacian controller to maintain relative positions in a formation. Studies in [63] found that the line formation is the optimal topology while the robot group moves in the cross-wind direction. Considering the 3-D OSL problem (i.e., the odor source is not on the ground), Soares *et al.* [64] proposed a 3-D triangle formation, where three ground vehicles maintain a line formation and an aerial robot moves behind the ground flock.

2.4.2 Swarm-based Algorithms

While robots need to maintain relative positions in a formation-based algorithm, swarm-based algorithms approach the coordination problem from another angle: a single robot has the autonomy to self-plan its trajectory rather than being treated as a part of the entire formation [65].

Hayes *et al.* [66] proposed a multi-agent odor search algorithm based on the anemotaxis method. Six robots were deployed in a closed environment with an electrical fan blowing wind from the right upper corner to the left downward corner (Fig. 2.7(a)). Each robot was implemented with the anemotaxis method (surge/casting) to search for the source. A histogram (Fig. 2.7(b)) containing the accumulated number of plume hits received from all robots was created, and the area with the highest bin number was the estimated source location.



(a) (b)

FIGURE 2.7: An experiment setup and a source likelihood map based on a multi-robot odor search strategy proposed by Hayes *et al.* [66].

Among various swarm-based algorithms, particle swarm optimization (PSO) [67] is a commonly used multi-agent algorithm in an OSL task. Inspired by the behavior of bird flocks randomly searching for food, PSO is a computational algorithm that optimizes a problem via iteratively improving a candidate solution with regard to a given fitness (i.e., objective) function. When applied in an OSL task, measured odor concentrations

(e.g., [68, 69]) or the probability of an area containing the odor source [70–72] can be utilized to define the fitness function. The best solution can be regarded as the possible odor source location. At each iteration, the best solution of a robot and the fleet are evaluated according to the given fitness function. Then, robot positions are updated to approach the best solutions.

Marques et al. [68] presented an odor search algorithm with multiple robots based on the PSO algorithm. At each time step, the robot with the highest detected odor concentration shares its position with others, and other robots will adjust their velocities and positions to approach it. After the possible odor locations are narrowed, the robots perform a closed search, such as the spiral search, to determine the odor source location. Meng et al. [70] proposed a variant of the original PSO method to coordinate multiple robots: the fitness function was selected as the area that gives the highest odor source probability rather than the highest odor concentration.

A primary shortcoming of the standard PSO (S-PSO) is that robots can be trapped in local optima which are not the odor source location. To address this problem in an OSL task, several variant PSO algorithms were introduced. Jatmiko *et al.* [73] presented the detection and responding PSO, where a random spread is performed when the global best solution has not changed for a period of time. A similar idea can be found in [74], where robots are prevented from moving too close, ensuring more widespread exploration. Yan *et al.* [75] proposed a 'request and reset' PSO, which requests robots with low fitness leave their current positions and resets the rest of the robots to search any possible areas with higher fitness values.

Another drawback is that S-PSO does not use the airflow information, which is considered as a useful cue for finding an odor source. Feng *et al.* [76] proposed a modified

PSO algorithm, which includes an upwind term in updating robot velocities since odor plumes released from the source will be advected to the downstream areas. Another method proposed in their work [77] adds a random disturbance term along with an upwind term to update robot velocities. Therefore, robots prefer upwind movements in the plume tracing process, improving the search efficiency.

2.4.3 Summary of Multi-agent Search Algorithms

By summarizing these works, it can be found that implementing a multi-agent system in an OSL task can improve the search efficiency compared to a single-agent system. With the increasing number of robotic agents, the distributed sensing information at multiple locations and the cooperation between the robots can greatly increase the search efficiency.

The effective coordination of robots to collaboratively search for the odor source is central to a multi-agent OSL algorithm. Simply adding some single-agent algorithms to the multi-agent case will not guarantee a performance improvement since robots may hinder each other during the search. Derived from this consideration, formation-based and swarm-based coordination algorithms are designed to organize robot search behaviors, where formation-based algorithms control robots as a whole unit and robots in swarm-based algorithms can self-plan their search trajectories.

It should be mentioned that simply adapting existing multi-agent algorithms, e.g., formation control or PSO, for the OSL task is not ideal since they are not originally designed for the OSL problem. For instance, the PSO-based OSL algorithm does not consider airflow information in guiding robots to detect plumes. This operation is not ideal for efficiently locating the odor source since the airflow information indicates the

direction the plume is coming from, which can be used to deduce possible odor source locations. Minor modifications, such as adding an upwind term in the PSO algorithms to force robots to select upwind movements [76, 78], cannot fundamentally solve this problem. Therefore, more sophisticated and intelligent collaboration strategies that are specifically designed for the OSL task are required to improve the search efficiency.

Chapter 3

Adaptive Bio-inspired Navigation Using Fuzzy Inference Methods

This chapter presents an adaptive bio-inspired plume tracing algorithm using fuzzy inference methods. The objective is to design an olfactory-based navigation algorithm that compromises benefits from both bio-inspired and engineering-based methods. To achieve this, the simple and concise framework of a bio-inspired method is utilized to reduce the algorithm complexity and absorb the intellectual ability (i.e., the ability to mathematically estimate environmental changes) from engineering-based methods to improve the search performance. Therefore, this method is constructed based on the 'surge/casting' behavior framework and enhanced with a decision-making approach via the fuzzy inference theory to perceive the environment and adjust robot search trajectories according to the current search situation.

3.1 An Overview of the Proposed Method

As mentioned, an OSL task can be separated into three search phases, namely plume finding, plume tracing, and source declaration [9]. The first search phase, i.e., plume finding, aims to verify the existence of plumes in the search area. After the robot detects plumes for the first time, the plume tracing phase is activated, in which the robot detects plumes as cues to approach the odor source. Once the robot gains enough information to confirm the source position, it declares the estimated source location in the source declaration phase, which is also considered as the end of an OSL task.

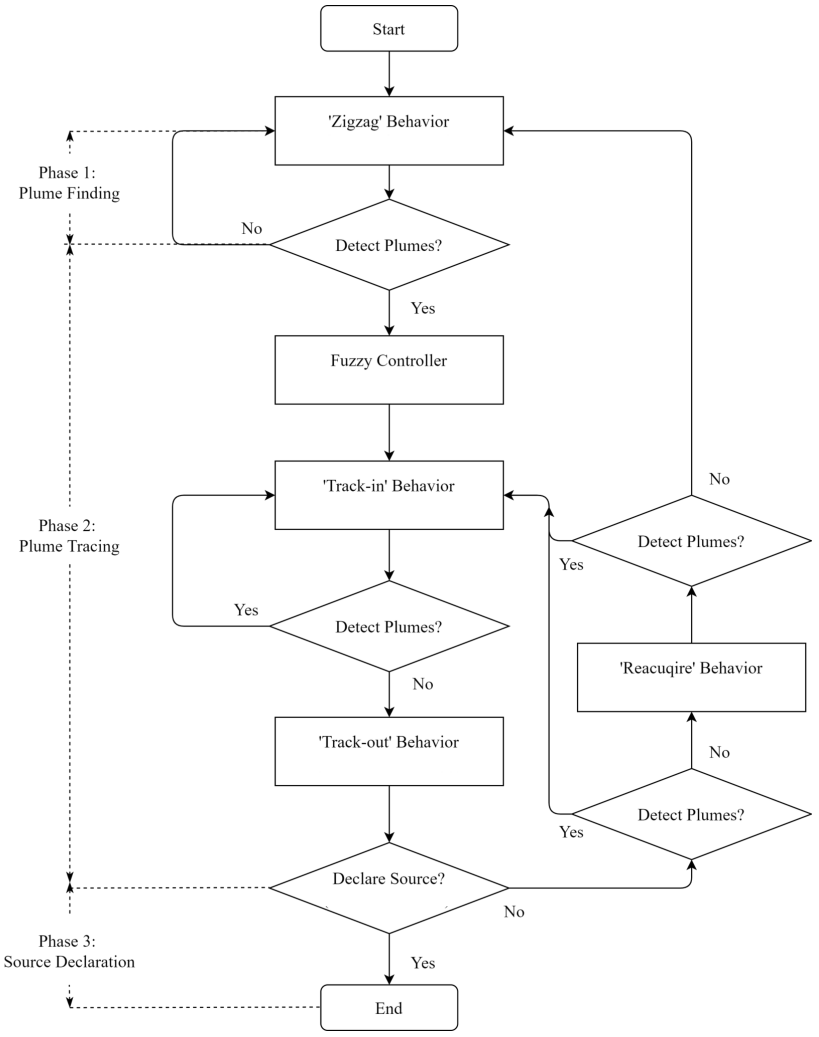


FIGURE 3.1: The flow diagram of the proposed olfactory-based navigation algorithm

In this method, inspired by male moths' mate-seeking behaviors, multiple search behaviors are designed to implement in the aforementioned search phases. Fig. 3.1 presents the flow diagram of these search behaviors in the proposed algorithm. Initially, the 'zigzag' search behavior is employed in the plume finding phase, which guides the robot to detect plumes for the first time. After the initial plume detection, the robot switches to the plume tracing phase, which consists of three search behaviors, namely 'track-in', 'track-out', and 'reacquire'. The designed fuzzy controller is activated at the beginning of the plume tracing phase.

In the plume tracing phase, the 'track-in' behavior is activated once the robot detects plumes. Similar to the 'surge' behavior of male moths, the 'track-in' behavior tries to make rapid progress toward the odor source while plumes have been detected. When the robot moves out of plumes, the 'track-out' behavior is triggered to manipulate the robot to traverse the plume trajectory. The hope is that the robot can encounter plumes via the 'track-out' behavior, but if it does not, then the 'reacquire' behavior is activated. Like the 'casting' behavior of male moths, the robot in the 'reacquire' behavior performs crosswind excursions to detect plumes over a wide region. If the robot still cannot detect plumes, it will switch back to the plume finding phase and repeat the 'zigzag' behavior. Once the robot detects plumes, it turns back to the plume tracing phase and performs the 'track-in' behavior to trace plumes in the upwind direction.

In the source declaration phase, the robot examines whether it can confirm a source location after the 'track-out' behavior. If the robot collects enough information and is confident with the source estimation, it declares the estimated source location and completes the OSL task.

3.2 Behavior-Based Navigation Algorithm

In this section, a bio-inspired navigation algorithm based on [33] and [37] is presented, which is designed inspired by mate-seeking behaviors of male moths. It should be mentioned that the OSL task defined in this work is a two-dimensional (2-D) problem since the aimed implementation platform is a ground robot.

3.2.1 Plume Finding

The plume finding phase aims to verify the existence of plumes within the search area. Without any assumptions about the odor source location, the crosswind search is more efficient than the along-wind search to detect plumes since the robot is more likely to encounter plumes while moving in the crosswind direction [79]. Thus, the robot search trajectory is dominant with the crosswind movements and includes a smaller along-wind component to ensure the exploration.

Derived from these considerations, we design and implement the 'zigzag' behavior in this search phase. Fig. 3.2 presents a sample 'zigzag' search trajectory. It can be observed that the main features of this behavior include: 1) the robot moves predominantly across the airflow direction; 2) the trajectory also contains an along-wind component to cause the robot to explore new regions; 3) when the robot reaches boundaries, it turns the heading toward the inside of the search area to continue the search. While the robot maneuvers, the plume tracing phase is activated at any time when the sensed odor concentration exceeds the detection threshold, which terminates the current 'zigzag' search behavior.

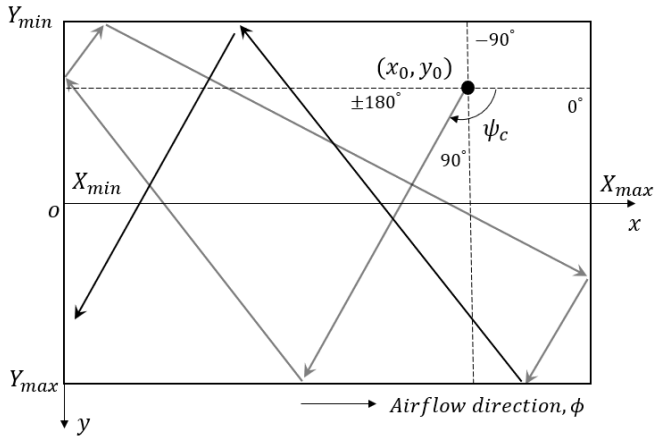


FIGURE 3.2: The sample 'zigzag' search trajectory in the plume finding phase, where X_{min}, X_{max}, Y_{min}, and Y_{max} represent boundaries of the search area in the horizontal and vertical directions, respectively. The current robot location is at (x_0, y_0), and the commanded heading is ψ_c.

3.2.2 Plume Tracing

The objective of the plume tracing phase is to command the robot to approach the odor source via tracing emitted plumes. Three search behaviors are designed and implemented in this search phase, including 'track-in', 'track-out', and 'reacquire'.

3.2.2.1 'Track-in' Behavior

Inspired by the 'surge' behavior of male moths, the robot in the 'track-in' behavior is commanded to move upwind when it detects odor plumes. Studies in [80] reveal that immediately following a plume detection, a good plume tracing performance is attained by driving at an offset angle $\beta \in [20°, 70°]$ relative to the upwind direction. The benefit of this design is that when the robot drives out of plumes with a nonzero offset angle β, it can predict which side of plumes it exited from and perform a counter-turn to re-contact plumes.

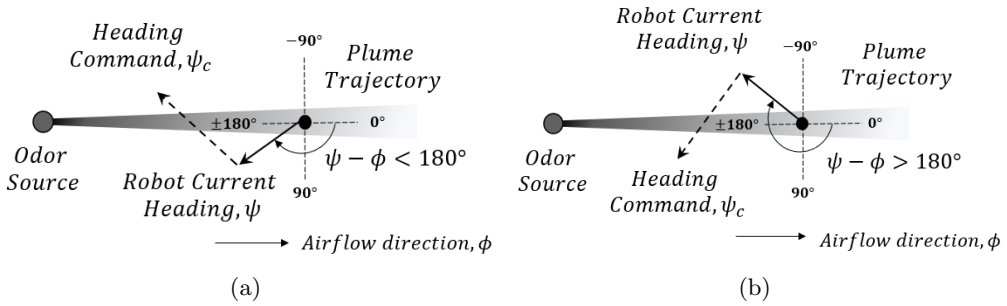


FIGURE 3.3: Demonstrations for determining the value of LHS in the 'track-in' behavior, where (a) $LHS = 1$ and (b) $LHS = -1$.

To realize this mechanism, the heading command ψ_c in the 'track-in' behavior, which is positively defined in clockwise rotations and 0° is at the positive x axis, is presented as:

$$\psi_c = \phi + 180 + LHS \cdot \beta, \tag{3.1}$$

where ϕ is the sensed airflow direction at the robot position, and LHS (i.e., stands for the phrase: "*L*eft *H*and *S*ide") is an indicator that is either ± 1. The value of LHS reflects the side of plumes that the robot would drive out, which is determined based on the difference between the robot heading angle ψ and the airflow direction ϕ, i.e., $\psi - \phi$. As presented in Fig. 3.3, if the robot drives out of plumes from the left side of the plume trajectory (when looking upwind), i.e., $(\psi - \phi) < 180°$, the value of LHS is defined as 1; otherwise, the robot is expected to leave plumes from the right side, i.e., $(\psi - \phi) > 180°$, and the value of LHS is -1. In either case, the calculated heading command directs the robot to perform a counter-turn from the side that it drives out of the plumes and remains inside the plume trajectory.

Algorithm 1 presents the pseudo-code for the 'track-in' behavior. When the robot detects plumes, i.e., the sensed odor concentration exceeds the threshold, the heading command ψ_c is updated via (3.1), and the plume detection time T_{last} and position $\mathbf{P}_{last}$ are recorded. When plumes are absent, the robot will hold the current heading for λ

Algorithm 1 'Track-in' Behavior

Set the speed command $v = v_c$
if sens.Odor $\geqslant$ threshold **then**
    Determine the value of LHS:
    if $(\psi$ - $\phi) < 180$ **then**
        $LHS = 1$
    else
        $LHS = -1$
    end if
    Update the heading command ψ_c:

$$\psi_c = \phi + 180 + LHS \cdot \beta$$

    Record the last detection time T_{last} and position $\mathbf{P}_{last}$:

$$T_{last} = t;\ \mathbf{P}_{last} = (x, y)$$

else
    if $(t - T_{last}) > \lambda$ **then**
        Save $\mathbf{P}_{last}$ into a list $\mathcal{LP}$:

$$\mathcal{LP}[i] = \mathbf{P}_{last};\ i++$$

        return 'Track-out' Behavior
    end if
end if
return 'Track-in' Behavior

seconds to confirm the plume non-detection event. If the robot cannot detect plumes within λ seconds, i.e., $(t - T_{last}) > \lambda$ (t is the current time), the latest last plume detection location $\mathbf{P}_{last}$ will be added to a "last detection position" list, which is named as $\mathcal{LP}$. Then, the robot switches to the 'Track-out' behavior and inhibits the current 'track-in' behavior.

3.2.2.2 'Track-out' Behavior

The 'track-out' behavior attempts to make progress toward the odor source and quickly re-detect plumes in the vicinity of the last plume detection location. To achieve these two objectives, the robot is commanded to move toward a target point $\mathbf{P}_{target}$ that is

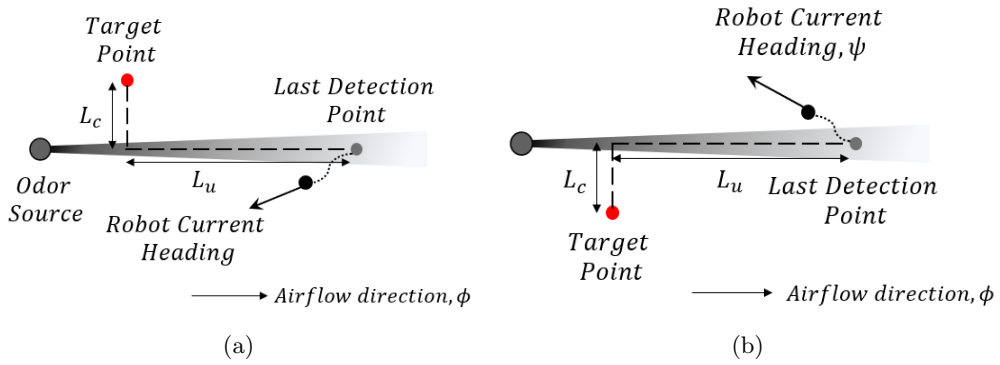


FIGURE 3.4: Demonstrations for determining the target position $\mathbf{P}_{target}$ (painted in red) in the 'track-out' behavior, where (a) $LHS = 1$ and (b) $LHS = -1$.

L_u meters upwind and L_c meters crosswind from the most upwind position in the "last detection position" list, i.e., $\mathcal{LP}$.

Algorithm 2 presents procedures in the 'track-out' behavior. A critical step is to determine the correct crosswind direction: by following this direction, the robot is expected to traverse the plume trajectory and re-detect plumes. To achieve this mechanism, the indicator LHS, which predicts the side of plumes that the robot drives out, is employed to calculate $\mathbf{P}_{target}$. As presented in Fig. 3.4, if the robot leaves plumes from one side, the calculated target position will be located at the opposite side of plumes. Therefore, by proceeding to the target position, the robot is expected to move across the plume trajectory and re-detect plumes.

The 'track-out' behavior is terminated either when the robot detects plumes or reaches the target point. In either case, the robot checks whether it can declare the odor source location before determining the next behavior (i.e., `SourceCheck` in Algorithm 2): if a source location can be declared, the robot switches to the source declaration behavior and completes the OSL task; if plumes are detected but the source cannot be declared, the robot turns to the 'track-in' behavior; if plumes are not detected and the source cannot be declared, the robot switches to the 'reacquire' behavior to search plumes over

Algorithm 2 'Track-out' Behavior

Set the speed command, $v = v_c$
if sens.Odor >= threshold **then**
    if `SourceCheck` **then**
        return 'Source declaration' Behavior
    else
        return 'Track-in' Behavior
    end if
else
    Determine the target position, $\mathbf{P}_{target}$:
        1) Find the upwind point $\mathbf{P}_{up}$ in $\mathcal{LP}$
        2) Find the unit flow vector, $\mathbf{F} = (\cos\phi, \sin\phi)$
        3) Rotate $\mathbf{F}$ clockwise by 90° to get $\mathbf{F}_p$
        4) $\mathbf{P}_{target} = \mathbf{P}_{up} - L_u \cdot \mathbf{F} - L_c \cdot LHS \cdot \mathbf{F}_p$
    Calculate the heading command:

$$\psi_c = \arctan\left(y_{target} - y\right)/\left(x_{target} - x\right)$$

    if |sens.VehPosition - $\mathbf{P}_{target}$| $< R$ **then**
        if `SourceCheck` **then**
            return 'Source declaration' Behavior
        else
            return 'Reacquire' Behavior
        end if
    end if
end if
return 'Track-out' Behavior

a larger scale.

Notice that, values of L_u and L_c decide the scale of search trajectories in the 'track-out' behavior. Similar to parameters defined in the 'track-in' behavior, L_u and L_c are also adjusted by the fuzzy controller. When the airflow becomes turbulent, a small value of L_u and a large value of L_c will be generated to emphasize the crosswind component in the search trajectory, which increases the likelihood of re-detecting plumes in turbulent environments. On the other hand, a large value of L_u and a small value of L_u will be produced to enhance the upwind movement in laminar flow environments, which brings the robot further close to the odor source and saves the search time.

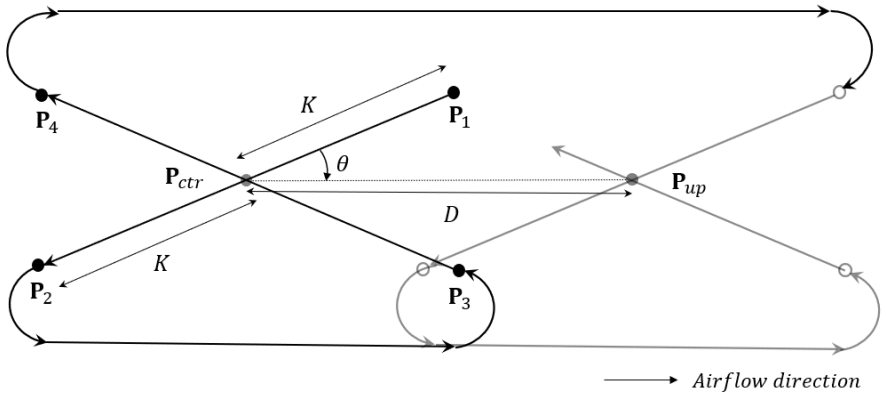


FIGURE 3.5: The Bow-tie trajectory used in the 'reacquire' search behavior. The robot starts to perform the first Bow-tie centered in $\mathbf{P}_{center}$, and if the robot completes this Bow-tie without plume detection, it repeats the Bow-tie centered in $\mathbf{P}_{up}$. The distance between two Bow-ties is D meters.

3.2.2.3 'Reacquire' Behavior

The 'reacquire' behavior is designed to restore the plume contact in the situation where the robot fails to detect plumes in the 'track-out' behavior.

Biological studies [81, 82] suggest that when a male moth loses contact with pheromone plumes, it ceases the upwind movement and progressively performs crosswind excursions. Inspired by this behavior, the Bow-tie trajectory is designed and implemented in the 'reacquire' behavior. This trajectory enables the robot to move across the airflow direction multiple times to increase the likelihood of re-detecting plumes. As shown in Fig. 3.5, a Bow-tie trajectory is constructed based on a center point $\mathbf{P}_{ctr}$, and by defining a Bow-tie radius K and an offset angle θ, four corner points ($\mathbf{P}_1$ to $\mathbf{P}_4$) are determined around $\mathbf{P}_{ctr}$. The robot will reach each corner point in ascending order to complete a Bow-tie trajectory. Besides, the robot turning radius is considered while designing the transition trajectory between two horizontal points.

If the robot fails to detect plumes in a Bow-tie, it will repeat this trajectory one more time to find plumes. As presented in Fig. 3.5, the initial Bow-tie trajectory is centered in $\mathbf{P}_{ctr}$, which is D meters away from the most upwind point $\mathbf{P}_{up}$ in the last detection

Algorithm 3 'Reacquire' Behavior

```
Set the speed command, v = v_c
if sens.Odor < threshold then
    Find the most upwind point in LP, P_up
    Find the unit flow vector, F = (cos φ, sin φ)
    P_ctr = P_up − D · F
    Perform the 'Bow-tie' trajectory at P_ctr and P_up
    if BowTie(P_ctr and P_up) == done then
        Remove the current P_up from LP
        if LP is empty then
            return 'Zigzag' Behavior
        end if
    end if
else
    return 'Track-in' Behavior
end if
return 'Reacquire' Behavior
```

position list, i.e., $\mathcal{LP}$. The next Bow-tie trajectory is centered at $\mathbf{P}_{up}$, and if the robot completes these two Bow-ties without plume detection, the current $\mathbf{P}_{up}$ will be removed from $\mathcal{LP}$. The robot then repeats the 'reacquire' behavior at the most upwind point on the remaining points in $\mathcal{LP}$. The 'reacquire' behavior is terminated when plumes are detected, which switches the robot to the 'track-in' behavior, or $\mathcal{LP}$ becomes empty, which reverts the robot to the plume finding phase, i.e., the 'zigzag' behavior.

3.2.3 Source Declaration

The robot identifies and declares the odor source location in the source declaration phase. Unlike previous search behaviors, there is no clear analog in animal olfactory behaviors to the robot declare-source behavior. For instance, in the mate-seeking behaviors of male moths, the final determination of a female moth location could be completed based on information from multiple perceptions, which may include vision, tactile, and auditory cues [33]. However, in an OSL problem, the robot is required to determine the

odor source location based only on a series of plume detection events, which is more engineering-oriented than biologically inspired.

In this work, the spatial distribution of the last plume detection positions is utilized to declare the odor source location. As mentioned, the last plume detection positions are recorded at the end of 'track-in' behaviors. Since the odor source location is fixed, when the robot is far from the odor source, the last detection positions will be widely separated along the airflow direction; when the robot is close to the odor source, the last detection positions will be densely accumulated at the downflow area near the odor source location. This feature can be utilized to generate the source declaration algorithm, which includes two steps:

1. Recorded last detection positions are sorted with respect to the airflow direction, i.e., the most upwind point is placed at the beginning, then is the second upwind point, and so on;

2. When the first ϵ last detection positions differ in the airflow direction by less than γ meters, the most upwind point (i.e., the first point) is declared as the odor source location.

Values of ϵ and γ are determined such that the trade-off between the source declaration accuracy and the processing time is balanced. Generally, increasing the number of points used in the source declaration decision, i.e., ϵ, and decreasing the distance between consecutive points, i.e., γ, may increase the accuracy and reliability of the estimated source location but may also increase the time required to satisfy the declaration criteria. In implementations, to obtain a well-algorithm performance and save the processing time, ϵ and γ are defined as 3 and 2, respectively.

3.3 Design of the Fuzzy Inference System

3.3.1 Design Concept

Parameters in search behaviors, including β and λ in the 'track-in' behavior, L_u and L_c in the 'track-out' behavior, and K, θ, and D in the 'reacquire' behavior, determine the scale of robot trajectories and the transition between different search behaviors. Traditional bio-inspired methods, such as [33] and [37], assign values of these parameters by trial and error prior to the search. Once parameter values are settled, they are fixed during the entire search, which is not an ideal setting for real-world environment with varying airflow fields.

An improved design is to adjust search parameters according to the current search situation dynamically. For instance, when the airflow becomes turbulent, odor plumes are advected by strong winds and meander in a wide range. In this scenario, large values of β, L_c, K, and θ are preferred to generate an oscillating search trajectory covering a large search area, which is beneficial for detecting meandering plumes over a wide range. Conversely, when the airflow is laminar, values of λ, L_u, and D should be large to produce a smooth search trajectory, which is desired to keep the robot maintaining inside a stable plume trajectory. Besides the airflow characteristics, the distance between the robot and the odor source is also critical to affecting values of search parameters: if the robot is near the source, it should search locally to exploit the exact odor source location, i.e., values of parameters should be small to constrain the scale of search trajectories; otherwise, the robot should explore the search area over a broad region to collect the odor source information, i.e., values of parameters should be large to extend the scale of search trajectories.

The challenging part of designing such a controller is that it is required to analyze the current search situation and transfer the changes of search situations to a quantitative mechanism that adjusts search parameters. Considering uncertainties in the source location and search environment, mathematical methods to quantitatively analyze differences in search situations are challenging to implement. Inspired by the implementations of fuzzy theory in the field of decision-making [83] and data classification [84], which successfully handles the problems with vagueness and uncertainties, a fuzzy inference approach is employed to perceive the environment and adjust parameters in search behaviors. In the fuzzy theory, vague variables and environments can be handled in a deterministic manner via linguistic descriptions and rules. Therefore, by analyzing sensor data, such as airflow measurements and odor concentrations, the current search situation is expected to be identified by the fuzzy controller. Then, values of behavior parameters can be adapted based on the defined fuzzy rules to achieve an optimal search performance.

3.3.2 Define Inputs and Outputs of the Fuzzy Controller

The block diagram of the proposed fuzzy controller is presented in Fig. 3.6. Inputs of the fuzzy controller are utilized to conjecture the current search situation, and outputs are coefficients that adjust values of behavior parameters. Search situations are categorized into four types with linguistic descriptions, namely laminar, turbulent, near the source, and far from the source. The first two search situations describe the search environment's airflow characteristics, and the last two indicate how close the robot is to the odor source.

There are three signals as the inputs of the fuzzy controller, namely turbulence intensity TI, sensed odor concentration at the robot position ρ, and the plume non-detection period δ_T, i.e., the period since the last detection event. Among inputs, turbulence

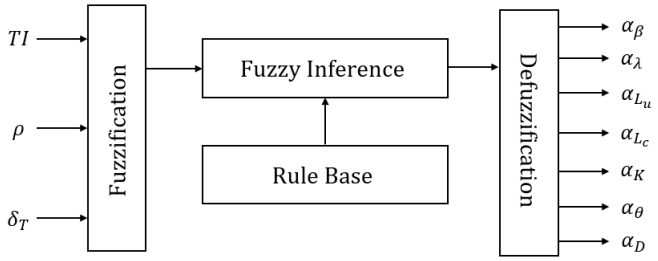


FIGURE 3.6: Schematic diagram of the proposed fuzzy logic controller. TI is the turbulence intensity of the search environment, ρ is the sensed odor concentration at the robot position, and δ_T is the non-detection period. α_β, α_λ, ... , α_D are coefficients that adjust behavior parameters.

intensity TI is a parameter that estimates airflow characteristics, which is defined as [85]:

$$TI = \frac{\sigma_u}{m_u}, \tag{3.2}$$

where σ_u and m_u are the standard deviation and the mean of airflow velocities u, respectively, and σ_u is calculated as [86]:

$$\sigma_u = \sqrt{\frac{1}{2}(\sigma_{u_x}^2 + \sigma_{u_y}^2)}, \tag{3.3}$$

where σ_{u_x} and σ_{u_y} are the standard deviation of airflow velocities on the x and y directions, i.e., u_x and u_y. Besides, m_u is computed from mean velocity components:

$$m_u = \sqrt{m_{u_x}^2 + m_{u_y}^2}, \tag{3.4}$$

where m_{u_x} and m_{u_y} are the mean of u_x and u_y. It can be observed that a larger TI indicates a higher level turbulence in the airflow environment.

In implementations, values of TI are computed based on the most recent airflow measurements within H seconds (the sampling frequency of the airflow sensor in implementations is 100 Hz). As shown in Fig. 3.7, various values of H have been evaluated (including H = 20, 100, and 200), and test results show that when H is small, TI is

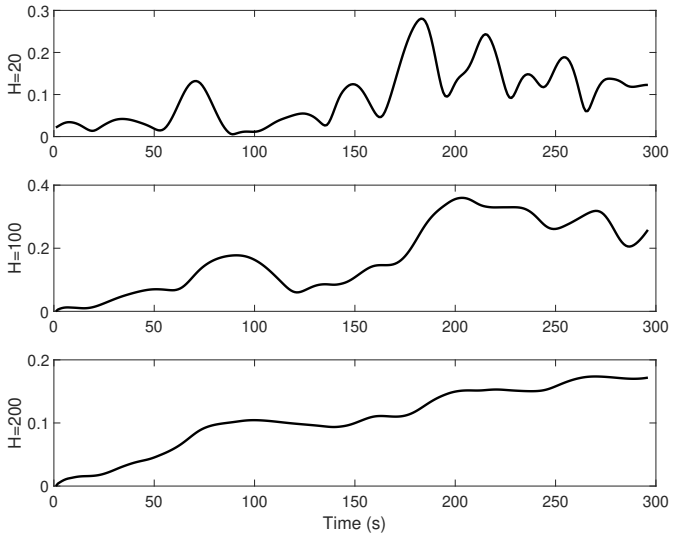


FIGURE 3.7: Plots of TI with different values of H in an OSL trial, where H is the size of the buffer that stores airflow measurements. From the top diagram to the bottom diagram, the values of H are 20, 100, and 200, respectively.

more sensitive to instantaneous airflow changes in the environment, and when H is large, TI is sluggish to the instantaneous airflow changes and becomes the averaged airflow value over a period. In this work, TI is expected to reflect the airflow changes in the robot surrounding environment timely; thus, based on test results, H is determined as 20 s in implementations.

The remaining two fuzzy inputs are the sensed odor concentration ρ and the plume non-detection period δ_T. Unlike traditional olfactory-based navigation algorithms, such as [37] and [51], which simplify the magnitude of ρ as a binary detector, this work utilizes the analog concentration signal to estimate the distance from the robot to the odor source. Notice that, due to the existence of local concentration maxima along the plume trajectory [87], a single high concentration detection is not enough to prove that the robot is near the odor source. Therefore, δ_T is added to the fuzzy inputs to assist the estimation. Since the positions of local concentration maxima are time-varying in turbulent flow environments [88], if the robot consecutively detects high odor

concentrations in a short period, i.e., ρ is high and δ_T is short, the robot is very likely to be near the source.

Outputs of the fuzzy controller are a group of coefficients α that adjust search behavior parameters, including α_β, α_λ, α_{L_u}, α_{L_c}, α_K, α_θ, and α_D. Each coefficient is applied on the base values of behavior parameters, such as $\beta = \alpha_\beta \cdot \beta_{base}$, $\lambda = \alpha_\lambda \cdot \lambda_{base}$, etc. The output range of all coefficients is from 0 to 1. Therefore, by varying values of coefficients, behavior parameter values will be changed correspondingly.

3.3.3 Procedures of the Fuzzy Controller

3.3.3.1 Fuzzification

The fuzzification is a procedure that maps the crisp input values to linguistic fuzzy terms with a membership value between 0 and 1, which represents the degree of uncertainty that input values belong in a fuzzy set.

Fig. 3.8 presents plots of membership functions and fuzzy sets for fuzzy inputs and outputs, and the Gaussian membership function is selected to implement on all fuzzy sets. Since the number of potential fuzzy rules depends on the size of fuzzy sets of input variables, small numbers of fuzzy sets are defined for each input: three fuzzy sets are defined for the turbulence intensity TI, namely laminar (La), averaged (Av), and Turbulent (Tu); two fuzzy sets are defined for the sensed odor concentration ρ, namely low (L) and high (H); two fuzzy sets are defined for the plume non-detection period δ_T, namely short (Sh) and long (Lo). For all fuzzy outputs (i.e., α_β, α_λ, ..., α_D), the fuzzification procedure is identical since the discourse of universe of each coefficient is equivalent, i.e., $\alpha \in [0, 1]$. Specifically, five fuzzy sets are defined over the discourse of

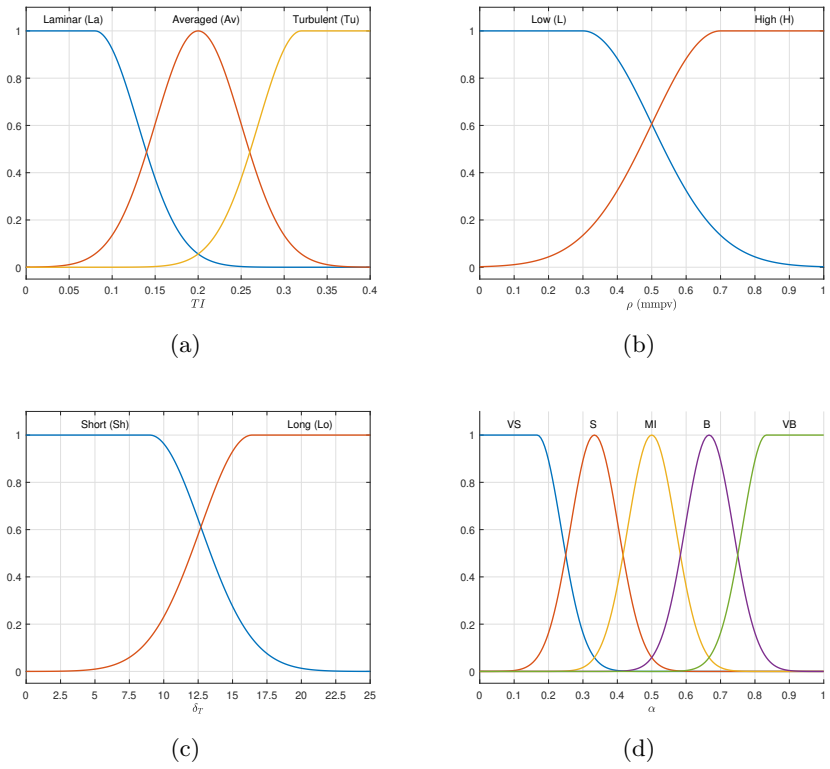


FIGURE 3.8: Membership functions and fuzzy sets for inputs and output of the fuzzy controller. Inputs include: (a) turbulence intensity TI, (b) sensed odor concentration ρ, and (c) plume non-detection period δ_T. The output is (d), which represents coefficients of behavior parameters, i.e., α_β, α_λ, ..., α_D.

universe of the output, namely very small (VS), small (S), middle (MI), big (B), and very big (VB).

3.3.3.2 Fuzzy Rules

Fuzzy rules are responsible for the decision making in a fuzzy controller, which govern the input-output relationship. Each rule is given by a "IF-THEN" statement [89], where the "IF" part defines the combination of fuzzy inputs, and the "THEN" part specifies the consequent results of fuzzy outputs.

In the proposed fuzzy controller, fuzzy rules stipulate how the robot reacts to different search situations. Here, we design fuzzy rules based on our previous research experiences in both bio-inspired and engineering-based plume tracing algorithms. We discover that

the search efficiency can be improved if the robot can extend its crosswind excursions to explore larger areas when it is in a turbulent environment and decrease the intensity of crosswind movements to save time if the robot is in a laminar flow environment. Borrowing this idea, we want the robot to explore (i.e., find plumes in a large area) if it is in a turbulent environment and exploit (i.e., find the odor source in a constraint area) when the surrounding airflow environment is laminar.

To achieve this mechanism, the characteristics of the airflow environment is estimated. If the environment is turbulent, values of α_β, α_{L_c}, α_K, and α_θ are increased to generate oscillating search trajectories to improve the exploration; otherwise, these values are decreased and α_λ, α_{L_u}, and α_D are increased to generate smooth trajectories that emphasize upwind movements, which help the robot to approach the odor source in a laminar environment quickly. Additionally, values of α are fine-tuned by the estimated distance between the robot and the odor source: if the robot is far from the odor source, the robot inclines to find plumes over a large area, i.e., exploration; otherwise, the robot tends to search the odor source within a constrained area, i.e., exploitation. Specifically, in the stage of exploration, values of α are increased to generate the large scale search trajectories, and in the stage of exploitation, values of α are constrained to limit the scale of search trajectories.

An example fuzzy rule is demonstrated as follow: when TI is laminar, ρ is high, and δ_T is short, the robot is expected to be close to the odor source in a laminar environment; thus, α_λ, α_{L_u}, and α_D are increased and α_β, α_{L_c}, α_K, and α_θ are decreased to render a smooth trajectory in the current laminar environment, and considering the close distance to the odor source, values of all coefficients should be constrained in a small scale to generate local search trajectories within a limited area. In the "IF-THEN" format, the above rule can be presented as:

TABLE 3.1: List of Fuzzy Rules. La: Laminar; Av: Averaged; Tu: Turbulent; L: Low; H: High; Sh: Short; Lo: Long; VS: Very Short; S: Short; MI: Middle; B: Big; VB: Very Big.

Rule	Inputs			Outputs						
No.	TI	ρ	δ_T	α_β	α_λ	α_{L_u}	α_{L_c}	α_K	α_θ	α_D
1	La	L	Sh	VS	B	B	VS	VS	VS	B
2	La	L	Lo	S	VB	VB	S	S	S	VB
3	La	H	Sh	VS	S	S	VS	VS	VS	S
4	La	H	Lo	VS	MI	MI	VS	VS	VS	MI
5	Av	L	Sh	B	MI	MI	B	MI	B	S
6	Av	L	Lo	B	MI	MI	B	B	B	S
7	Av	H	Sh	VS	VS	VS	S	VS	VS	VS
8	Av	H	Lo	S	VS	VS	S	VS	S	S
9	Tu	L	Sh	B	VS	VS	B	B	B	VS
10	Tu	L	Lo	VB	S	S	VB	VB	VB	S
11	Tu	H	Sh	S	VS	VS	S	S	S	VS
12	Tu	H	Lo	MI	VS	VS	MI	MI	MI	VS

$\mathcal{F}^1 = \{$IF TI is La AND ρ is H AND δ_T is Sh, THEN α_β is VS AND α_λ is S AND α_{L_u} is S AND α_{L_c} is VS AND α_K is VS AND α_θ is VS AND α_D is S.$\}$

Enumerate all possible combinations of inputs and outputs, Table 3.1 presents eighteen fuzzy rules in the proposed fuzzy controller.

3.3.3.3 Defuzzification

Defuzzification is a procedure that maps the fuzzy output to a crisp signal. In this work, the centroid method [89] is selected as the defuzzification algorithm, which can be expressed as follow:

$$\alpha = \frac{\sum_{i=1}^{n} Q_i \cdot \mu\left(Q_i\right)}{\sum_{i=1}^{n} \mu\left(Q_i\right)}, \tag{3.5}$$

where α is the coefficient value, which could be α_β, α_λ, ...; i is the index of fuzzy rules, i.e., $i \in [1, 18]$; Q_i denotes the center of the fired membership function of the output variable provided by the ith rule; $\mu(Q_i)$ is the output of the conjunction degree of the IF part of the ith rule.

3.4 Experiments and Results

In this section, we evaluate the effectiveness of the proposed olfactory-based navigation algorithm and compare the results with traditional bio-inspired [37] and engineering-based [51] methods. Considering the difficulty of repeatedly conducting OSL tests in the real-world environment, we employ a realistic plume tracing simulator as the evaluation tool. Based on our previous real-world experiment results [37, 90], the simulation based evaluation tool allows graphical and batch statistical analysis of plume tracing performance. It allows the user to analyze performance using either a simulated plume or data obtained from field or plume experiments. The simulation includes an evolving flow field, a coherent chemical plume with realistic short term chemical signatures and long term exposure, and a full six degree of freedom autonomous vehicle dynamics. Other research works, such as [91–95], also utilized this simulator as the evaluation platform.

3.4.1 Simulation Setup

3.4.1.1 Simulated Environment

Fig. 3.9 presents the simulated search area, where a coordinate $(x - y)$ is constructed to represent positions. An odor source is located at $(20, 0)$ m and releases 10 filament packages (i.e., plumes) per second. Released plumes form a circular plume trajectory as plotted by a grey-scale patchy trail. Arrows in the background represent airflow vectors, where the tail of an arrow points to the airflow direction and the length of an arrow indicates the strength of airflow velocity. In this simulator, airflow vectors are calculated from time-varying boundary conditions that are generated by a mean flow ($\mathbf{U}_0$) and Gaussian white noise (zero mean and ς variance). By changing values of

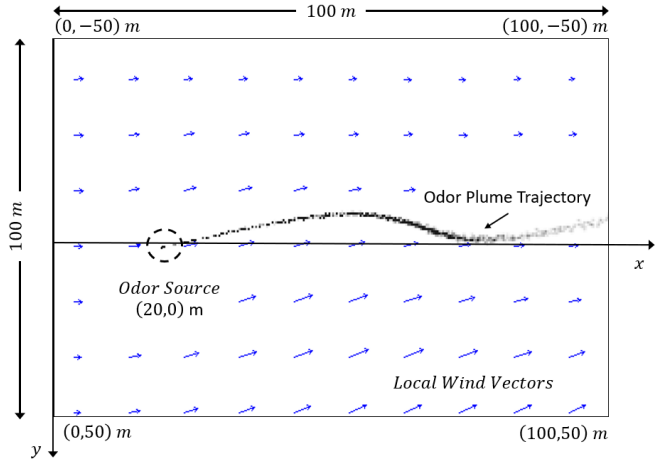


FIGURE 3.9: The simulated search area. The size of the search area is 100×100 m^2, and a fixed location odor source is placed at $(20, 0)$ m, which emits 10 plumes per second. Emitted plumes form a curvy trajectory as plotted by the grey-scale patchy trail. Airflow vectors, which are represented by blue arrows in the background, are calculated from a mean flow $\mathbf{U}_0$ and a variance ς. By changing these two variables, different airflow fields can be obtained.

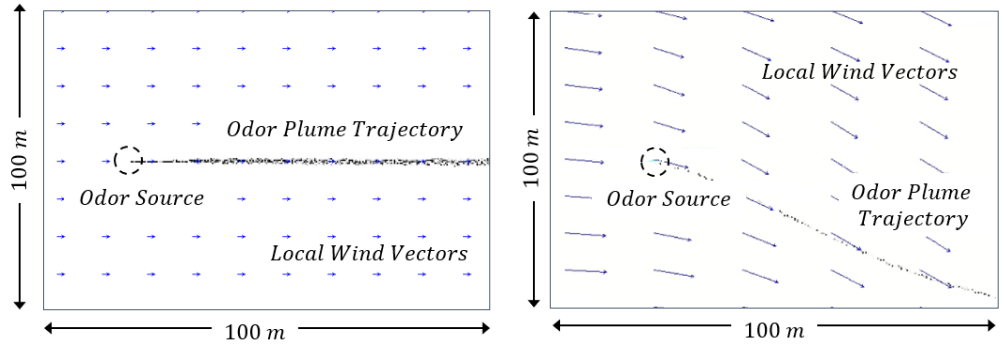


FIGURE 3.10: Airflow fields and corresponding odor plume trajectories in the simulation with different environmental settings. (a) Laminar Flows, $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 0$. (b) Turbulent Flows, $\mathbf{U}_0 = (3, 0.5)$ m/s and $\varsigma = 30$.

boundaries condition variables (i.e., $\mathbf{U}_0$ and ς), different amplitudes of airflow fields can be obtained.

Fig. 3.10 shows snapshots of two simulated airflow fields. In the left diagram, a laminar airflow environment is created with $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 0$, and in the right diagram, a turbulent airflow environment is generated with $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 10$.

3.4.1.2 Vehicle Assumptions

In the simulation program, a two-wheeled mobile robot is employed to implement the proposed navigation algorithm. Comparing to the large scale of the search area, the size of the robot is negligible. Therefore, the robot is approximated as a single point in the simulation program. It is assumed that the robot is equipped with a chemical sensor, an anemometer, and a positioning sensor, which measure odor concentrations, wind speeds and directions in the inertial frame, and the robot position in the inertial frame, respectively. All sensors' measurements are corrupted with Gaussian white noises to imitate real-world applications, where noise parameters are listed in Table 3.2. The proposed navigation algorithm is operated on an onboard computer to process sensor readings and calculate the heading and speed commands, which are limited in ranges of $\psi_c = [-180°, 180°]$ and $v_c = [0.6, 1]$ m/s, respectively.

The sampling frequency of all sensors is 100 Hz, while the implemented navigation algorithm produces a command per second. Considering olfactory sensing is characterized by very low false alarm rates but potentially high missed detection rates [16], we pick the highest chemical sensor reading during one decision-making period (i.e., 1 s) to identify whether the robot detects odor plumes. Here, a concentration threshold is employed to determine a detection event: if the sensed concentration exceeds the threshold, the detection event is confirmed; otherwise, the robot does not detect odor plumes. For other sensors, the averaged value of measurements among one decision-making period is fed to the navigation algorithm.

TABLE 3.2: Values of Parameters in Gaussian Noises in Sensor Measurements

	Chemical Sensor	Anemometer	Positioning Sensor
Mean	0	0	0
Standard deviation	0.05 mmpv[1]	0.1 m/s and 1°	0.1 m

[1] mmpv: million molecules per cm^3

3.4.1.3 Experiment Designs

Around 50 tests have been conducted in the simulation to evaluate the performance of the proposed olfactory-based navigation algorithm, which can be separated into three groups.

In the first group of tests, the effectiveness of the proposed navigation algorithm in a laminar flow environment is evaluated. Besides, robot trajectories in different search behaviors are demonstrated and compared with those generated without the designed fuzzy controller. Tests in the second group are carried out to investigate the validity of implementing the proposed navigation algorithm in a turbulent flow environment. Snapshots of robot trajectories at different time steps and the plots of fuzzy inputs and outputs are also presented. Tests in the last group are designed for evaluating the robustness of the proposed navigation algorithm, where various search conditions, including varying robot initial positions and airflow environments, are defined. Additionally, results of the proposed navigation algorithm in these tests are compared with traditional olfactory-based navigation algorithms.

3.4.2 Group 1: Implementation in a Laminar Flow Environment

In this group of tests, the robot is placed in a laminar flow environment, where $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 3$. The robot initial position is at $(80, -40)$ m and moves at a constant

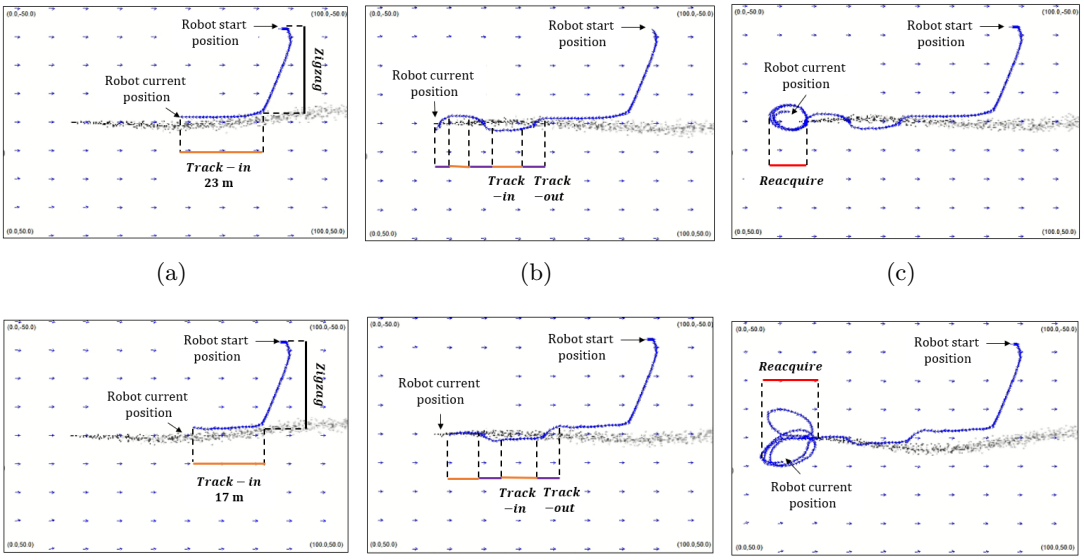


FIGURE 3.11: Snapshots of robot trajectories in the Group 1 tests. Among these diagrams, three top diagrams, i.e., (a), (b), (c), are robot trajectories generated with the proposed method, while the remaining diagrams, i.e., (d), (e), (f), are trajectories generated by the original bio-inspired method. Over these robot trajectories, durations of search behaviors are labeled by different color bars, where black is 'zigzag'; orange is 'track-in'; purple is 'track-out'; red is 'reacquire'. In Group 1 tests, the robot moves in a constant speed at 1 m/s. With the proposed method, the robot correctly locates the odor source with 175 s, which is much shorter compared to 351 s for the traditional bio-inspired counterpart.

speed 1 m/s. Fig. 3.11 presents robot trajectories in different search behaviors with and without the designed fuzzy controller.

At the initial phase of the search, the 'zigzag' trajectory is employed to guide the robot in detecting plumes. After the first plume detection event, the robot switches to the 'track-in' behavior. Comparing Fig. 3.11(a) and Fig. 3.11(d), it can be observed that with the proposed navigation algorithm, the trajectory length is longer than the counterpart in the 'track-in' behavior (23 m vs. 17 m). This is because the proposed fuzzy controller increases the value of λ in the 'track-in' behavior due to the laminar flow environment. This operation elongates the upwind movement, which helps the robot quickly approach the odor source.

Then, as shown in Fig. 3.11(b) and Fig. 3.11(e), the 'track-in' and 'track-out' behaviors

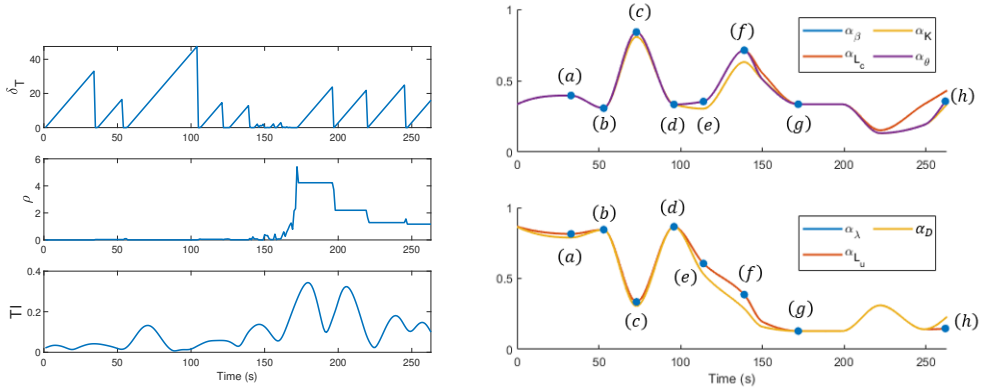
alternate to command the robot proceeding toward the odor source location. When the robot is close to the odor source, the fuzzy controller decreases the value of behavior parameters to constrain the scale of trajectories; therefore, the robot can perform a local search within a small area to exploit the exact odor source location. Comparing Fig. 3.11(c) and Fig. 3.11(f), which present robot trajectories in the 'reacquire' behavior, the trajectory with the fuzzy controller is more favorable since the robot circulates near the odor source location; by contrast, the trajectory without the fuzzy controller has a large scale, which is not efficient for saving the search time.

At the end of the search, the robot declares the odor source once the source declaration conditions are satisfied. The search time and the declared odor source location for the robot with the designed fuzzy controller is 175 s and $(21.7, 0.1)$ m, respectively. The distance to the real odor source location, i.e., $(20, 0)$ m, is 1.7 m. Considering the large size of the search area, i.e., 100×100 m^2, the performance of the proposed navigation algorithm is satisfied (within 5 m). Compared to the search results of the traditional bio-inspired method, where the robot uses 351 s and declares the odor source at $(21.2, 0.2)$ m, the proposed navigation algorithm is more efficient in Group 1 tests.

3.4.3 Group 2: Implementation in a Turbulent Flow Environment

In this group of tests, the proposed navigation algorithm is implemented in a turbulent flow environment, where $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 10$.

In Fig. 3.12, the change of fuzzy inputs and the corresponding fuzzy outputs, i.e., coefficient values, can be inspected and compared. When the airflow environment around the robot position is laminar, i.e., TI is at a low level, values of α_β, α_{L_c}, α_K, and α_θ, are low either, and values of α_λ, α_{L_u}, and α_A are high. Conversely, when the airflow

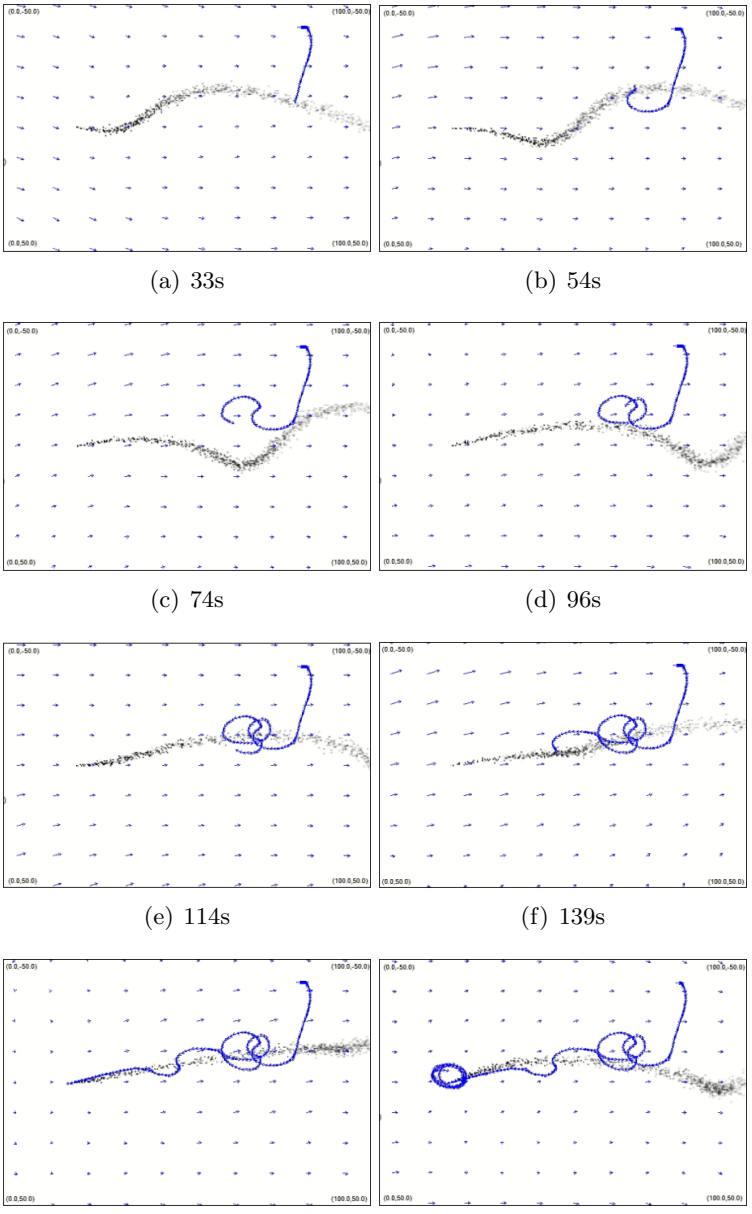


(a) (b)

FIGURE 3.12: Plots of fuzzy inputs and outputs generated in the Group 2 test, where (a) presents fuzzy inputs, and (b) shows fuzzy outputs. Labels on the fuzzy output plots, i.e., (a), (b), ... (h), are time steps that mentioned in Fig. 3.13.

environment becomes turbulent, i.e., TI grows to a high level, values of α_β, α_{L_c}, α_K, and α_θ increase and α_λ, α_{L_u}, and α_D decrease correspondingly. Besides, the shift of the robot from exploration to exploitation can also be observed. When the robot is near the odor source location, i.e., ρ is high and δ_T is short, values of all coefficients are constrained to produce a limited scale trajectory, which commands the robot to exploit the exact odor source location. Note that, the value of ρ will not be updated until the next above-threshold concentration measurement, where the threshold is defined to filter out the background concentration noises. Since the chemical sensor has a low false-alarm rate but a high miss-detect rate [16], this design enables the robot to memorize the most recent above-threshold concentration measurements, which helps the robot estimate the distance to the source.

Search details are demonstrated in Fig. 3.13. At $t = 33$ s, the robot encounters plumes for the first time and switches to the 'track-in' behavior. In the period from $t = 50$ s to $t = 75$ s, due to the turbulent airflows, the robot loses the plume contact and performs the 'track-out' and 'reacquire' behaviors to recover plumes. As presented in Fig. 3.13(c), the robot moves in circles to find plumes. Meanwhile, it can be observed in Fig. 3.12(b)



(a) 33s (b) 54s

(c) 74s (d) 96s

(e) 114s (f) 139s

(g) 172s (h) 262s

FIGURE 3.13: Snapshots of robot trajectories at different time steps in the Group 2 test. The robot is placed in a turbulent airflow environment with the environmental settings $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 10$. The robot starts at $(80, -40)$ m and declares the odor source location at 262 s. The declared odor source location is at $(20.2, 0.1)$ m, which is 0.22 m to the actual odor source location. It should be noted that the airflow field displayed at the background is at the current time while the shown plume trajectory is developed over a period.

that the fuzzy controller increases values of α_β, α_{L_c}, α_K, and α_θ and decreases α_λ, α_{L_u}, and α_D to generate the large scale trajectories for fostering the plume search over a large area. At around $t = 100$ s, the robot re-detects plumes and switches back to the 'track-in' behavior; however, this behavior does not last long due to the circulating plume trajectory produced by turbulent airflows. At $t = 139$, after performing a 'track-out' behavior to traverse the plume trajectory, the robot maintains inside plumes until $t = 172$ s. During this period, the robot quickly approaches the odor source location. In the plots of coefficients, i.e., Fig. 3.12(b), coefficient values are constrained to a low level after $t = 172$ s, which results in the small scale trajectories. It can be seen in Fig. 3.13(h) that the robot circulates around the odor source location with a small scale trajectory. At $t = 262$ s, the robot declares the odor source location, which is located at $(20.2, 0.1)$ m. The distance error to the actual odor source location is 0.22 m.

3.4.4 Group3: Comparisons to Traditional Methods

In this group of tests, the robustness of the proposed navigation algorithm in varying search conditions is evaluated. Two test scenarios have been designed as follows:

- Scenario 1: the robot starts the OSL task at different initial positions in a turbulent flow environment, where $\mathbf{U}_0 = (1, 0)$ m/s and $\varsigma = 8$.
- Scenario 2: the robot starts at the same initial position, i.e., $(80, -40)$ m, but airflow environments vary.

Additionally, to evaluate the performance of the proposed navigation algorithm, a traditional bio-inspired method [37] and an engineering-based method [15] are also implemented and compared in this group of tests. It should be mentioned that for each

navigation algorithm, the robot operates at the same speed, i.e., 1 m/s, and with the same source declaration algorithm (presented in Section 3.2.3).

3.4.4.1 Results of Scenario 1

Table 3.3 presents search results in Scenario 1 tests, where six different robot initial positions over the search area are evaluated.

TABLE 3.3: Search Times of Different Navigation Methods in Varying Robot Initial Positions

Test	Robot Initial Position (m)	Traditional Bio-inspired Method (s)	Traditional Engineering-based Method (s)	The Proposed Navigation Algorithm
1	(35, 20)	250	**138**	151
2	(50, 40)	279	**140**	215
3	(60, -40)	343	193	**189**
4	(70, 30)	345	164	**157**
5	(90, 40)	313	323	**195**
6	(90, -40)	399	467	**188**

For the proposed navigation algorithm, it can be observed in Table 3.3 that no matter how the robot initial position changes, the robot can correctly declare the odor source within 5 m to the actual source location. Compared to other two methods, the proposed method achieves shorter search time in Test 3-6, while for Test 1-2, the search time of the proposed method is comparable to the engineering-based counterpart. However, the time complexity of the implemented engineering-based method grows significantly with respect to the size of the search area, i.e., when the search area is large, the robot needs more time to generate a source probability map. The proposed algorithm, on the other hand, does not relate with the size of search area, where robot commands are calculated directly based on the sensors' measurements. Considering the long processing time of the implemented engineering-based algorithm, the proposed method is more preferable for implementations. Test results in Scenario 1 demonstrate the validity of the proposed navigation algorithm with varying robot initial positions.

TABLE 3.4: Search Results of Three Navigation Methods in Different Airflow Environments. f: Successful/Total Tests; μ: Averaged Search Time; σ: Standard Deviation of Search Time

Environmental Settings		Traditional Bio-inspired Method [37]			Traditional Engineering-based Method [15]			The Proposed Navigation Method		
$\mathbf{U}_0$ (m/s)	ς	f	μ (s)	σ	f	μ (s)	σ	f	μ (s)	σ
(1, 0)	8	20/20	336.5	57.0	18/20	297.1	65.7	**20/20**	**251.8**	**42.7**
(1, 0.5)	5	20/20	322.0	45.0	**20/20**	**227.2**	**36.0**	20/20	245.4	30.0
(1, 0)	10	20/20	326.7	32.8	20/20	300.9	78.1	**20/20**	**243.2**	**56.4**
(2, 0)	10	20/20	331.2	24.0	20/20	304.4	68.6	**20/20**	**252.8**	**36.8**
(2, 0.3)	15	20/20	349.5	40.8	20/20	334.8	63.2	**20/20**	**313.3**	**68.4**

TABLE 3.5: Statistical Results of Repeated Tests

	Successful/ Total Tests	Success Rate	Standard deviation of Search Time	Averaged Search Time (s)
Traditional Bio-inspired Method [37]	100/100	100%	42.5	333.2
Traditional Engineering-based Method [15]	98/100	98%	78.8	292.9
The Proposed Navigation Algorithm	100/100	**100**%	**55.6**	**260.7**

3.4.4.2 Results of Scenario 2

In Scenario 2, the proposed navigation algorithm is implemented in different airflow environments. For each navigation algorithm, the OSL trail is repeated 100 times to obtain the statistic results. Search results of all tests are presented in Table 3.4, and the statistical results are presented in Table 3.5.

Table 3.4 demonstrates that the proposed navigation algorithm outperforms the traditional bio-inspired method in all environments concerning the mean search time. Compared to the engineering-based method, the proposed algorithm achieves a shorter search time all environments except Env. 2, where the engineering-based method is slightly better than the proposed algorithm in terms of the mean search time (i.e., 227.2 s vs. 245.4 s). Table 3.5 presents the statistic results of three navigation algorithms. It can be seen that the proposed navigation algorithm achieves the shortest averaged search time among three navigation algorithms, which indicates that the proposed algorithm is more

efficient than the other two methods in these tests. As for the standard deviation of search time, the proposed method has a smaller value than the engineering-based method and comparable to the bio-inspired method, indicating that the proposed method is reliable in search performance. This result verifies the effectiveness of the proposed method, which enables the robot to exploit the exact odor source location when it is near the source. Considering the great computational demand of the engineering-based method, the proposed algorithm is more favorable to be implemented on mobile robots due to the low computation load and acceptable localization accuracy.

In general, the main advantage of the proposed navigation algorithm is that the computation complexity is much less, but the search performance is comparable to the engineering-based method [15]. The navigation method proposed in [15] contains a cell-based source mapping algorithm, which separates the search area into multiple small cells and computes the probability of each cell containing the odor source. For cell-based source mapping algorithms, the computational cost relates to the number of cells defined in the search area [16]. When we implement [15] in simulation, we defined the number of cells is 1000 considering the large size of the search area (i.e., 100×100 m^2). Therefore, at every time step, [15] repeats the probability calculation 1000 times. In comparison, the proposed navigation algorithm's computational complexity is fixed (i.e., it does not need to compute the cell-based probability). The robot is commanded based on sensor readings at the current time step, which is more reliable and much faster than the engineering-based counterpart in varying search environments.

3.5 Summary of the Chapter

In this chapter, an olfactory-based navigation algorithm for using on a mobile robot to find an odor source in an unknown environment is presented. Inspired by the mate-seeking behaviors of male moths, a behavior-based search framework is designed and constructed. To enable the robot perceive the environment and understand the current search situation, a fuzzy controller is designed and implemented to adapt parameters in search behaviors. As a result, when the airflow environment is turbulent, the robot trajectories are extended to allow the robot to find plumes over a large area, i.e., exploration. When the robot is near the odor source, search trajectories are limited to constrain the robot in a local search, i.e., exploitation. Experiment results show that the proposed navigation algorithm is valid in both laminar and turbulent flow environments. Compared to two traditional plume tracing methods, including a bio-inspired and an engineering-based methods, the proposed algorithm is more effective and efficient in terms of the averaged search time and success rates.

Chapter 4

Olfactory-Based Navigation via Model-Based Reinforcement Learning and Fuzzy Inference Methods

This chapter presents an intelligent olfactory-based navigation algorithm via reinforcement learning (RL) and fuzzy inference methods. This algorithm models the odor source localization (OSL) as an RL problem. During the odor plume tracing process, the belief state in a partially observable Markov decision process (POMDP) model is adapted to generate a source probability map that estimates possible odor source locations, and a hidden Markov model (HMM) is employed to produce a plume distribution map that premises plume propagation areas. Both source and plume estimations are fed to the robot, and a decision-making approach based on fuzzy inference is designed to dynamically fuse information from two maps and to balance the exploitation and exploration of

the search. After assigning the fused information to reward functions, a value iteration based path planning algorithm is presented to solve for the optimal action policy. Comparing to other commonly used olfactory-based navigation algorithms, such as moth-inspired and Bayesian inference methods, simulation results show that the proposed method is more intelligent and efficient.

4.1 Motivation and Research Niche

Comparing existing olfactory-based navigation strategies, the limitation of bio-inspired methods is that the robot lacks the capability of estimating odor plume locations. Thus, when odor plumes are not detected, the robot can only perform a time-consuming 'casting' behavior to recover plumes. As for engineering-based methods, if the robot is source seeking oriented (e.g. [14]), the search efficiency is not ideal since the source probability map is unreliable before the robot acquires enough odor source information. On the other hand, the search result is also not desired if the robot is plume seeking oriented (e.g. [56]) since it leans toward detecting plumes instead of locating the odor source. The research niche that our approach fits is to let the robot estimate both odor source and odor plume locations and fuse two estimations as the target to guide the robot. So that, not only does the robot search for the odor source location, but also it can quickly recover from plume non-detection events when it does not observe plumes.

Reinforcement learning (RL) algorithms are widely implemented in the field of artificial intelligence (AI). For instance, AlphaGo [96], an AI robot based on RL methods, defeated a couple of best professional human players in the game of Go. An RL algorithm models interactions between an agent and the environment: an agent receives rewards by performing actions, and the goal of the agent is to take the action that maximizes

the cumulative reward [97]. The framework of an RL algorithm is similar to an OSL problem: an agent could be considered as a robot that aims to find an odor source in an unknown environment. By appropriately defining reward functions, the robot is driven to choose actions that are beneficial to locate an odor source. The optimal policy is adapted as a search path that leads the robot to the maximal reward location.

In this chapter, an olfactory-based navigation method for using on a ground mobile robot is presented. The proposed method contains two main procedures, i.e., modeling and planning. In the modeling procedure, odor source and plume estimations are obtained. Specifically, belief states in a POMDP model are adapted to represent a source probability map, from which the robot estimates possible odor source locations. Besides, a plume distribution map that predicts odor plume propagation areas is obtained from a HMM based method. A fuzzy inference approach is designed to dynamically fuse the information from two maps, and the combined information is assigned to reward functions. In the planning procedure, a search route is determined based on the generated reward functions. The value iteration method is adopted to solve the RL problem and produces the optimal policy, which is a search route that leads the robot to the maximum reward location, i.e., the location that contains the most odor source information. Modeling and planning procedures are repeated until reward functions converge, which is considered as the complete of an OSL problem.

4.2 RL Basics

Before diving into the actual odor search algorithms, the author first introduces some basic concepts and terminologies used in RL algorithms. Notations used in this section follow Sutton's book [97].

4.2.1 Markov Decision Process

A Markov decision process (MDP) is a commonly used math model in RL algorithms. All states in an MDP model has "Markov" property, which means that the future is independent of the past given the present. In other words, the future only depends on the current state, not the historical trajectory, and the current state encapsulates all the information we need to decide the future. The property can be represented as:

$$\mathbb{P}\left[S_{t+1}|S_t\right] = \mathbb{P}\left[S_{t+1}|S_1, \ldots, S_t\right]. \tag{4.1}$$

A MDP model consists of five elements: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, and each element is defined as:

- $\mathcal{S}$ is a set of states;
- $\mathcal{A}$ is a set of actions;
- P is a transition probability;
- R is a reward function;
- γ is a discounting factor for future rewards.

Fig. 4.1 shows a MDP model within one time step. At time step t, the agent is in a state ($S_t = s$, $s \in \mathcal{S}$) and after it takes an action ($A_t = a$, $a \in \mathcal{A}$) to arrive in the next state ($S_{t+1} = s'$, $s' \in \mathcal{S}$), the agent will obtain a reward ($R_{t+1} = r$, $r \in \mathcal{R}$). The MDP and agent together give a sequence starting from the initial state S_1 and ending with the terminal state S_T.

The sequence $S_1, A_1, R_2, S_2, A_2, \ldots, S_T$ is termed episode, which describes how the agent interacts with the environment. The probability of transitioning from the current state

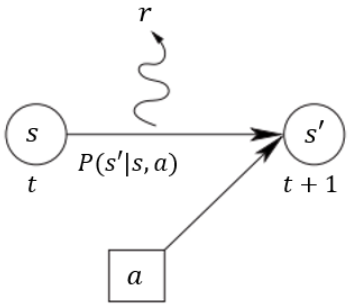


FIGURE 4.1: A process of a MDP model in a flow diagram within one time step. Reprinted from [98], Fig. 1.1.

(S_t) to a new state (S_{t+1}) after taking an action (A_t) and receiving a reward (R_{t+1}) is defined as $P(s', r|s, a)$:

$$P\left(s', r|s, a\right) = \mathbb{P}\left[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\right], \tag{4.2}$$

and

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} P\left(s', r|s, a\right) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}. \tag{4.3}$$

Thus, the state-transition function can be defined as:

$$P^a_{ss'} = P\left(s'|s, a\right) = \mathbb{P}\left[S_{t+1} = s'|S_t = s, A_t = a\right] = \sum_{r \in \mathcal{R}} P\left(s', r|s, a\right). \tag{4.4}$$

The reward function R predicts the next reward triggered by one action:

$$R(s, a) = \mathbb{E}\left[R_{t+1}|S_t = s, A_t = a\right] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} P\left(s', r|s, a\right). \tag{4.5}$$

4.2.2 Policy

The policy (π) describes which action the agent will take in the state S_t with the goal to maximize the total rewards. It is a mapping from state S_t to action A_t (i.e., a 'guidebook'

that indicates the agent to perform an action in a state) and can either be deterministic or stochastic:

- Deterministic: $\pi(s) = a$.
- Stochastic: $\pi(a|s) = \mathbb{P}_\pi[A_t = a|S_t = s]$.

A policy fully defines the behaviour of an agent, and due to the 'Markov' property, policies only depend on the current state, not on the history.

4.2.3 Value Functions

Value function is a prediction of the future reward. The future reward G_t, also known as a return, is a total sum of discounted rewards from the time step t, and it is represented as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{4.6}$$

where $\gamma \in [0, 1]$ is the discounting factor that penalizes rewards in the future. There are two kinds of value functions: state-value function and action-value function. The state-value function (i.e., $V_\pi(s)$) evaluates the goodness of the agent being in a state s, and the action-value function (i.e., $Q_\pi(s, a)$) evaluates the goodness of the agent performs an action a in a state s. The term of 'goodness' is defined in terms of future rewards that can be expected or estimated, i.e., the higher expected future rewards, the better the agent being in a state or performing a action in a state. Mathematically, $V_\pi(s)$ is the expectation of the return (total future rewards) starting from the current state S_t, and then following the policy π: $V_\pi(s) = \mathbb{E}_\pi [G_t|S_t = s]$; $Q_\pi(s, a)$ is the expectation of the return starting from state s, taking the action a, and then following the policy π:

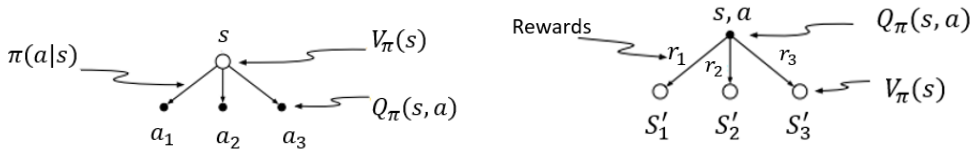
$Q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$. $V_\pi(s)$ can be represented by $Q_\pi(s,a)$ through:

$$V_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) Q_\pi(s,a). \tag{4.7}$$

Reversely, $Q_\pi(s,a)$ can also be represented by $V_\pi(s)$ through:

$$Q_\pi(s,a) = \sum_{s',r} P\left(s',r|s,a\right)\left[r + \gamma V_\pi\left(s'\right)\right] = R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{ss'} V_\pi\left(s'\right). \tag{4.8}$$

Backward root diagrams are helpful to understand these transformations. In Fig. 4.2(a), open circles represent states and solid circles represent state-action pairs. Starting from state s, the root node at the top, the agent could take one of three actions based on its policy π. To calculate $V_\pi(s)$, all $Q_\pi(s,a)$ are averaged with weights of action occurring probabilities. A similar root diagram is shown in Fig. 4.2(b). Starting from a state-action pair s,a, the agent could reach one of three states s' based on the transition probabilities $P(s',r|s,a)$ and receive a reward r corresponding with the state it selected. To calculate $Q_\pi(s,a)$, total rewards (immediate reward r plus future reward $V(s')$) are averaged with weights of transition probabilities.



(a) $Q_\pi(s,a) \Rightarrow V_\pi(s)$ (b) $V_\pi(s) \Rightarrow Q_\pi(s,a)$

FIGURE 4.2: Root diagrams of the state-value function and the action-value function

4.2.4 Bellman Equations

There are two kinds of Bellman equations, namely Bellman expectation equations and Bellman optimality equations. Bellman expectation equations provide a recursive way to calculate value function. For state-value function, the Bellman expectation equation is represented as:

$$V_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \left(R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{ss'} V_\pi\left(s'\right) \right), \tag{4.9}$$

and for action-value function, the recursive form is given as:

$$Q_\pi(s,a) = R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{ss'} \sum_{a'\in\mathcal{A}} \pi\left(a'|s'\right) Q_\pi\left(s',a'\right). \tag{4.10}$$

On the other hand, Bellman optimality equations are used to select the best action that gives the maximum value function. The Bellman optimality equation for a state-value function is shown as:

$$V_*(s) = \max_{a\in\mathcal{A}} \left(R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{ss'} V_*\left(s'\right) \right), \tag{4.11}$$

and for action-value function, it is represented as:

$$Q_*(s,a) = R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{ss'} \max_{a'\in\mathcal{A}} Q_*\left(s',a'\right), \tag{4.12}$$

where V_* and Q_* represent the maximum expectation of the return. The significance of Bellman optimality equations is that the optimal policy (π_*) can be obtained by choose actions that gives the maximum return: $\pi_* = \arg\max_a Q_\pi(s,a)$.

4.2.5 Solve RL Problems: Dynamic Programming

If the model is fully known to the agent, i.e., the agent knows reward functions and transition probabilities, the RL problem can be solved by using dynamic programming (DP) to iteratively evaluate value functions and improve the policy. A DP comprises two methods: policy iteration and value iteration.

Policy iteration methods provide an iterative procedure to improve the policy. There are three steps in a policy iteration method:

1. **Policy Evaluation (Prediction)**

   Compute the state-value function $V_{\pi,t+1}$ for a given policy π based on the state-value $V_{\pi,t}$ at the current time t using the Bellman expectation equation:

$$V_{\pi,t+1}(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \left(R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{ss'} V_{\pi,t}\left(s'\right) \right); \tag{4.13}$$

2. **Policy Improvement**

   Generate a better policy π' at time $t+1$ by acting greedily with respect to $V_{\pi,t+1}$, i.e., pick the action that gives the highest state-value function:

$$\pi' = \arg\max_{a\in\mathcal{A}} \left(R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{ss'} V_{\pi}\left(s'\right) \right); \tag{4.14}$$

3. **Check Convergence**

   If state-value function converges, then stop and return the optimal policy π^*; otherwise go back to policy evaluation.

Value iteration is the modification of the policy iteration. In this algorithm, the agent finds the best state-value function immediately trough the Bellman optimality equation

(Eqn. 4.11) without doing the policy evaluation step:

1. **Initialization**

   For all $s \in \mathcal{S}$, arbitrarily except that $V(terminal) = 0$;

2. **Bellman Optimality Equation**

   Find the best state-value function (one of these two):

   - Synchronous backup: $V_{k+1}(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} V_k\left(s'\right) \right)$

     At time step t, the agent calculates the next state-value function using the current state-value function. After all states have new state-value functions, update them synchronously.

   - Asynchronous backup: $V(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} V\left(s'\right) \right)$

     Don't wait all states get update, the agent calculates a new state-value function using whatever values of other states happened to be available and assigns the new state-value function to the state immediately. No matter

3. **Check Convergence**

   If the state-value function converges, then stop and output π^*; otherwise go back step 2.

4.3 Overview of the Proposed Olfactory-based Navigation Method

The scheme of the proposed olfactory-based navigation method is presented in Fig. 4.3. In the plume finding phase, a 'zigzag' search route presented in [33] is adopted to detect plumes for the first time, and the source mapping algorithm that estimates odor source locations (see Section 4.4.3.5) is activated simultaneously. After the robot

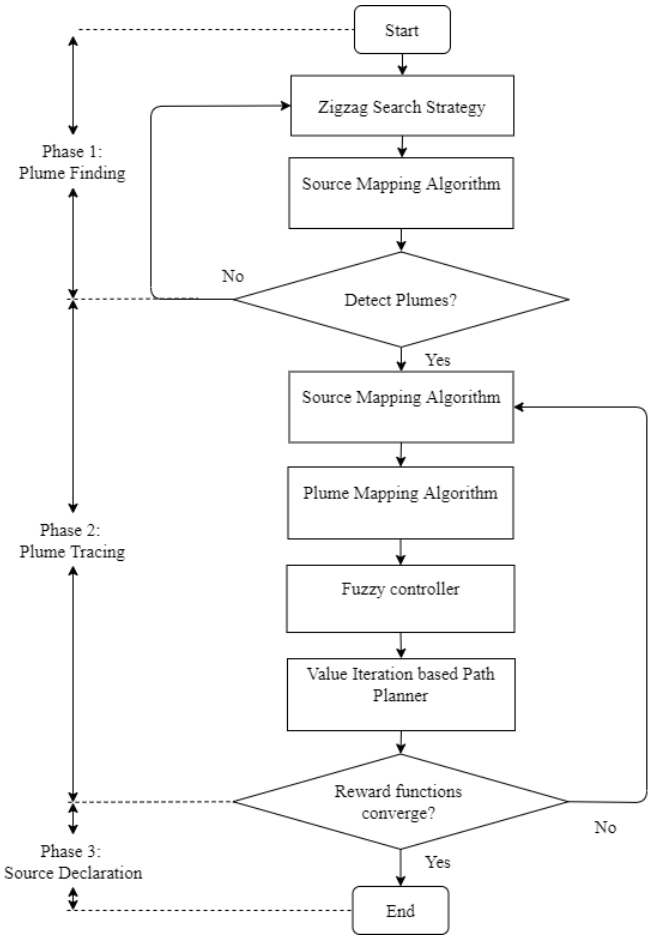


FIGURE 4.3: The framework of the proposed olfactory-based navigation method

detects plumes for the first time, the robot switches to the plume tracing phase, in which the source mapping algorithm sustains and the plume mapping algorithm that predicts plume propagation areas (see Section 4.4.4) is activated. Results from source and plume mappings are combined via a fuzzy controller to form reward functions (see Section 4.5.1). Then, a search route is generated by the value iteration based path planning algorithm (see Section 4.5.2). In the plume tracing phase, aforementioned algorithms keep updating until reward functions converge, which indicates that the robot finds the odor source location.

Specifically, the proposed method can be separated into two principle procedures, namely modeling and planning. In the modeling procedure, source estimates are obtained from belief states in a POMDP. The motivation of using belief states is that under the POMDP

framework, belief states can estimate uncertainties of an environment in a stochastic fashion, i.e., the agent cannot directly observe states, but it can estimate the current state through the belief state, which is the probability of the agent being in a state. To adapt the POMDP framework in an OSL problem, the odor source location can be used to define hidden states since it is unknown to the robot, actions can be considered as possible moving directions of the robot, observations can be adapted as plume detection and non-detection events, and the belief state can be interpreted as the probability of an area containing the odor source, i.e., a source probability map. Besides, to construct a plume distribution map, which estimates plume propagation areas, a HMM based method is adopted.

In the planning procedure, robot searching routes are determined. In an RL algorithm, reward functions determine behaviors of an agent and stipulate how we want the agent to accomplish its objective. In the proposed method, reward functions are expected to contain the information that not only conjectures odor source locations but also estimates plume propagation areas since source and plume estimations are instructive for the robot to either exploit or explore the odor source location. Thus, a source probability map and a plume distribution map are combined to form reward functions. Instead of combining them in a fixed proportional pattern (e.g., [99]), a fuzzy controller is designed to identify the current search condition and dynamically balance weights of two maps (exploitation or exploration). Then, a value iteration based path planning algorithm is adopted to solve for the optimal policy, i.e., the optimal search route that leads the robot toward the location containing the most odor source information.

4.4 Modeling

4.4.1 Search Area

In this work, the OSL is considered as a two-dimensional (2-D) problem since the aimed implementation platform of the proposed olfactory-based navigation method is a ground mobile robot. For computational feasibility, the search area is modeled as a rectangular grid with m cells in a row and n cells in a column as shown in Fig. 4.4. The size of a cell is defined as $L_x \times L_y$, where L_x and L_y are the length and width of a cell in the x and y directions respectively. A vector $\mathbf{C} = [C_1, C_2, ..., C_M]$ is defined to store cell indexes, where $M = mn$. Besides, C_i can also represent the position of a cell, such that $C_i = (x_i, y_i)$ is the center point of a cell C_i, $i \in [1, M]$. The odor source is placed in one of M cells, and its location is clouded to the robot.

Let $f \in [1, m]$ count over cells in the x direction and $g \in [1, n]$ count over cells in the y direction. Then, a cell index $i \in [1, M]$ can be represented as $i = f + (g-1)m$. Reversely, if i is given, f and g can be calculated as:

$$\begin{aligned} f(i) &= \text{rem}(i-1, m) + 1 \\ g(i) &= \text{int}(i-1, m) + 1, \end{aligned} \tag{4.15}$$

where $\text{rem}(n, m)$ and $\text{int}(n, m)$ are the remainder of n being divided by m and the greatest integer that is less than or equal to n/m, respectively. Thus, two cell representations, i.e., C_i and $C_{(f,g)}$, are equivalent.

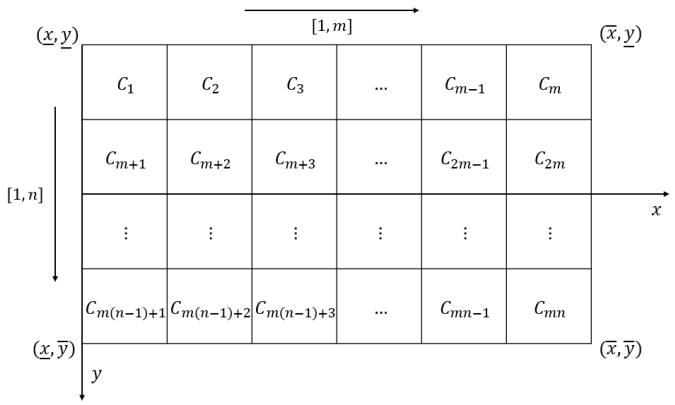


FIGURE 4.4: The search area defined in the OSL problem

4.4.2 Probabilities of Detecting and Not Detecting Plumes

4.4.2.1 Plume Model

In a turbulent flow environment, movements of odor plumes follow a random walk superimposed on the airflow advection, which can be expressed as [51]:

$$\dot{\mathbf{X}}(t) = \mathbf{U}(\mathbf{X}, t) + \mathbf{N}(t), \tag{4.16}$$

where $\mathbf{X} = (x_p, y_p)$ is the odor plume location, $\mathbf{U} = (u_x, u_y)$ is the mean wind velocity, which transports a plume as the whole body (i.e., advection), and $\mathbf{N} = (n_x, n_y)$ denotes the random walk velocity, which stirs filaments inside a plume and changes the plume shape (i.e., diffusion). $\mathbf{N}$ can be modeled as a Gaussian random process with zero mean and $\boldsymbol{\sigma}^2 = (\sigma_x^2, \sigma_y^2)$ variance, where σ_x and σ_y are strengths of the random walk velocity in the x and y directions, respectively. It should be noted that the position of an odor plume is chiefly determined by the advection $\mathbf{U}$, since the strength of mean wind velocity is much higher than that of the random walk velocity [36].

If the odor source releases a single odor plume at time t_l, the location of the odor plume at time t_k ($t_l < t_k$) can be calculated by integrating (4.16):

$$\mathbf{X}\left(t_l, t_k\right) = \mathbf{X}_s\left(t_l\right) + \int_{t_l}^{t_k} \mathbf{U}(\mathbf{X}(\tau), \tau) d\tau + \int_{t_l}^{t_k} \mathbf{N}(\tau) d\tau, \tag{4.17}$$

where $\mathbf{X}_s(t_l)$ is the odor source location. If the odor source continuously releases plumes in the time interval $[t_l, t_k]$, assuming that the release rate is G plumes per second, there are $G(t_k - t_l)$ plumes released, and positions of released plumes can be denoted by $\mathbf{P}(t_l, t_k) = [\mathbf{X}(t_l), \mathbf{X}(t_l + d\tau), \mathbf{X}(t_l + 2d\tau), ..., \mathbf{X}(t_k)]$ where $d\tau = 1/G$.

4.4.2.2 Single Released Odor Plume

This section presents the probability of detecting a plume in an arbitrary cell given that the source only releases a single plume.

Let $\mathbf{s}(t_l, t_k) = (s_x, s_y) = \int_{t_l}^{t_k} \mathbf{U}(\mathbf{X}(\tau), \tau) d\tau$, where s_x and s_y are plume advection distances in the x and y directions, respectively. Let $\mathbf{W}(t_l, t_k) = (w_x, w_y) = \int_{t_l}^{t_k} \mathbf{N}(\tau) d\tau$, which is a Gaussian random process with zero mean and $(t_k - t_l)\boldsymbol{\sigma}^2$ variance. Thus, (4.17) can be rewritten as:

$$\mathbf{X}(t_l, t_k) = \mathbf{X}_s(t_l) + \mathbf{s}(t_l, t_k) + \mathbf{W}(t_l, t_k). \tag{4.18}$$

It is worth mentioning that $\mathbf{s}(t_l, t_k)$ is approximated by integrating the sensed airflow velocities at the robot position from t_l to t_k since the global airflow information is absent: $\mathbf{s}(t_l, t_k) = (s_x, s_y) \approx \sum_{q=l}^{k-1} \mathbf{u}(\mathbf{X}_v(t_q), t_q) dt$, where $\mathbf{u}(\mathbf{X}_v(t), t)$ denotes airflow measurements at the location $\mathbf{X}_v(t)$ (i.e., the robot location at time t). This approximation will

introduce additional errors, but, since the global airflow information is unavailable, this assumption is acceptable.

If the odor plume is propagated to a cell C_j at the time t_k, the estimated odor source location $\hat{\mathbf{X}}_s = (x_s, y_s)$ can be obtained by solving (4.18):

$$\hat{\mathbf{X}}_s(t_l, t_k) = \mathbf{X}(t_k) - \mathbf{s}(t_l, t_k) - \mathbf{W}(t_l, t_k), \tag{4.19}$$

where $\mathbf{X}(t_k) = (x_j, y_j)$ is the plume location at time t_k, which is inside the cell C_j. Note that, since $\mathbf{X}(t_k) - \mathbf{s}(t_l, t_k)$ is a constant and $\mathbf{W}(t_l, t_k)$ is a Gaussian random variable with zero mean and $(t_k - t_l)\boldsymbol{\sigma}^2$ variance, $\hat{\mathbf{X}}_s$ is also a Gaussian random variable with $\mathbf{X}(t_k) - \mathbf{s}(t_l, t_k)$ mean and $(t_k - t_l)\boldsymbol{\sigma}^2$ variance. Thus, the probability density function (PDF) of $\hat{\mathbf{X}}_s$ in the x and y directions are:

$$\begin{aligned} f(x_s) &= \frac{e^{-\frac{(x_j - s_x - x_s)^2}{2(t_k - t_l)\sigma_x^2}}}{\sqrt{2\pi\,(t_k - t_l)\,\sigma_x^2}}, \\ f(y_s) &= \frac{e^{-\frac{(y_j - s_y - y_s)^2}{2(t_k - t_l)\sigma_y^2}}}{\sqrt{2\pi\,(t_k - t_l)\,\sigma_y^2}}. \end{aligned} \tag{4.20}$$

Since x and y directions are orthogonal, the joint PDF of $\hat{\mathbf{X}}_s$ is $f(x_s) \times f(y_s)$. Then, the probability of the estimated odor source being located in an arbitrary cell can be calculated by integrating the PDF over positions in that cell.

Let $p_{ij}(t_l, t_k)$ donate the probability of there being an odor source in a cell C_i that released a single odor plume at time t_l given that the odor plume is in the cell C_j at

time t_k. Thus, $p_{ij}(t_l, t_k)$ can be calculated as:

$$
\begin{aligned}
& p_{ij}\left(t_l, t_k\right) \\
&= \int_{x_s \in C_i} \int_{y_s \in C_i} \frac{e^{-\frac{(x_j - s_x - x_s)^2}{2(t_k - t_l)\sigma_x^2}}}{\sqrt{2\pi(t_k - t_l)\sigma_x^2}} \frac{e^{-\frac{(y_j - s_y - y_s)^2}{2(t_k - t_l)\sigma_y^2}}}{\sqrt{2\pi(t_k - t_l)\sigma_y^2}} dx_s dy_s \\
&= \int_{x_i - \frac{L_x}{2}}^{x_i + \frac{L_x}{2}} \int_{y_i - \frac{L_y}{2}}^{y_i + \frac{L_y}{2}} \frac{e^{-\frac{(x_j - s_x - x_s)^2}{2(t_k - t_l)\sigma_x^2}} e^{-\frac{(y_j - s_y - y_s)^2}{2(t_k - t_l)\sigma_y^2}}}{2\pi\left(t_k - t_l\right)\sigma_x \sigma_y} dx_s dy_s \\
&= \int_{-\frac{L_x}{2}}^{\frac{L_x}{2}} \int_{-\frac{L_y}{2}}^{\frac{L_y}{2}} \frac{e^{-\frac{(x_j - x_i - s_x - x_s)^2}{2(t_k - t_l)\sigma_x^2}} e^{-\frac{(y_j - y_i - s_y - y_s)^2}{2(t_k - t_l)\sigma_y^2}}}{2\pi\left(t_k - t_l\right)\sigma_x \sigma_y} dx_s dy_s,
\end{aligned}
\tag{4.21}
$$

which is a function of the relative positions of the cell C_j (i.e., plume position) and the cell C_i (i.e., possible odor source position), plume advection distances $\mathbf{s}(t_k, t_l)$, and the plume propagation time $t_k - t_l$. In the algorithm implementation, $p_{ij}(t_l, t_k)$ is approximated as

$$
p_{ij}(t_l, t_k) = \frac{e^{-\frac{(x_j - x_i - s_x)^2}{2(t_k - t_l)\sigma_x^2}} e^{-\frac{(y_j - y_i - s_y)^2}{2(t_k - t_l)\sigma_y^2}}}{2\pi(t_k - t_l)\sigma_x \sigma_y} L_x L_y \tag{4.22}
$$

for the calculation efficiency. This approximation will introduce additional errors when the cell size is large (i.e., $L_x >> \sigma_x$ and $L_y >> \sigma_y$), but in this work, the cell size is small (i.e., $L_x \approx \sigma_x$ and $L_y \approx \sigma_y$), thus, the produced errors are negligible.

One feature of the olfactory sensing device is that it has trivial false-alarm rates, but high missed-detection rates [51]. To model this mechanism, let β donate the probability of the robot successfully detecting plumes given that there are detectable plumes at the chemical sensor position. Thus, the probability of detecting a single released plume in C_j at t_k that was released from C_i at t_l is $\beta p_{ij}(t_l, t_k)$, and the probability of not detecting this plume is $1 - \beta p_{ij}(t_l, t_k)$.

4.4.2.3 Continuous Released Odor Plumes

If the odor source continuously releases odor plumes from t_l to t_k, what is the probability of detecting and not detecting odor plumes? To answer this question, the value of t_l should be clarified.

As the plume traveling time $t_k - t_l$ increases, the value of $\mathbf{s}(t_l, t_k)$ grows correspondingly, but $\mathbf{s}(t_l, t_k)$ should not exceed the size of the search area since the robot cannot detect odor plumes outside of it. The value of t_l is initialized as 0 because the airflow velocity is unavailable before the search, and as the search progresses (i.e., t_k increases), the distance between t_l and t_k should be constrained (i.e., $t_k - t_l < h$, where h is the maximum length of recorded flow history) to satisfy the aforementioned restraint. Therefore, t_l is defined as:

$$t_l = \max(0, t_k - h + 1). \tag{4.23}$$

In Section 4.4.2.2, the probability of not detecting a plume in a cell C_j at time t_k due to a single odor plume release from a cell C_i at time t_l is calculated as $1 - \beta p_{ij}(t_l, t_k)$. Thus, if all release times within $[t_l, t_k]$ are accounted, the probability of not detecting plumes in a cell C_j at time t_k due to the continuous plume release from a cell C_i is:

$$\kappa_{ij}\left(t_l, t_k\right) = \prod_{t_l}^{t_{k-1}} \left[1 - \beta p_{ij}\left(t_l, t_k\right)\right]. \tag{4.24}$$

Since plume detection and non-detection events are complementary, the probability of detecting plumes under the same condition (i.e., continuous plume release) is $1 - \kappa_{ij}(t_l, t_k)$.

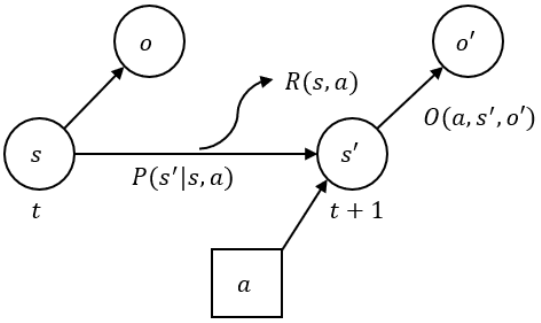


FIGURE 4.5: A basic POMDP model

4.4.3 Source Mapping

Belief states in a POMDP model are adapted as source estimations, which are used to construct a source probability map. A basic POMDP model can be defined by a tuple $(\mathcal{S}, \mathcal{A}, \Omega, P, O, R, b_0)$ as shown below [98]:

- $\mathcal{S}$ is a state space;
- $\mathcal{A}$ is an action space;
- Ω is an observation space;
- P are state transition probabilities between states;
- O are observation probabilities;
- R is the reward function defined on the transitions;
- b_0 is an initial probability distribution over states.

As shown in Fig. 4.5, at each time-step, the agent receives an observation (o, $o \in \Omega$) at the current state (s, $s \in \mathcal{S}$), and after performing an action (a, $a \in \mathcal{A}$), the agent is transferred to a new state (s', $s' \in \mathcal{S}$) according to the state transition probability $P(s'|s, a)$ and receives a new observation (o', $o' \in \Omega$) with the observation probability $O(a, s', o')$ and a reward $R(s, a)$. Since states are hidden to the agent, a probability

distribution over states is defined as the belief state $b(s)$, which indicates the probability of the agent being in a particular state s, and the initial belief state is b_0.

To illustrate the proposed source mapping algorithm, the rest of section presents an approach that adapts elements in a basic POMDP model to the context of an OSL problem.

4.4.3.1 State Space

States in a basic POMDP model are hidden to the agent, i.e., the agent does not know which state it is in. In an OSL problem, the actual odor source location is unknown to the robot. Thus, we defined states as the actual odor source location, which can be represented by a length-M (M is the number of cells in the search area) vector of Boolean values indicating whether each cell contains the odor source. If the Boolean value is 1, then the corresponding cell contains the odor source; otherwise, this value is 0.

For instance, $s_1 = [1, 0, ..., 0]$ indicates that the odor source is in the cell C_1, $s_2 = [0, 1, ..., 0]$ represents that the odor source is in the cell C_2, etc. Since the odor source could be located in an arbitrary cell inside the search area, the state space is represented as $\mathcal{S} = \{s_1, s_2, ...s_M\}$.

4.4.3.2 Action Space

The action space defines possible actions that an agent could select. As shown in Fig. 4.6, at the center cell $C_{(f,g)}$, the robot could select one of eight actions and enter the corresponding cell around it.

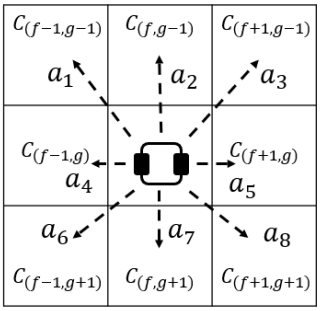


FIGURE 4.6: Action space. The robot location is at the center cell. Arrows indicate possible actions that the robot can take.

In this work, an action is represented by the destination cell. For example, a_2 in Fig. 4.6 can be represented as $a_2 = C_{(f,g-1)}$ since the destination cell of this action is $C_{(f,g-1)}$. Thus, the action space can be represented as $\mathcal{A}$= $\{a_1 = C_{(f-1,g-1)}, a_2 = C_{(f,g-1)}, ..., a_8 = C_{(f+1,g+1)}\}$.

4.4.3.3 State Transition Probabilities

The location of the odor source is stationary in this work, i.e., the odor source cannot move. Thus, the state transition probability is 1 if the new state is the same as the old state; otherwise, this probability is 0:

$$P(s' = s_i | s = s_j, a) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}, \tag{4.25}$$

where $i, j \in [1, M]$.

4.4.3.4 Observation Space and Probabilities

When the robot enters a cell C_j, it could or could not detect odor plumes. Thus, two observation states are defined in the observation space $\Omega = \{d, \bar{d}\}$, namely the plume detection event d and the plume non-detection event $\bar{d}$. A fixed plume concentration

threshold is adopted to identify two events, i.e., a plume detection event is confirmed when the sensed concentration is higher than the threshold, otherwise, a plume non-detection event is confirmed.

When the robot enters a cell C_j, the probability of detecting continuous released plumes is $1-\kappa_{ij}(t_l,t_k)$ and the probability of not detecting these plumes is $\kappa_{ij}(t_l,t_k)$ as defined in Section 4.4.2.3. Thus, the observation probability $O(a,s',o')$ is defined as:

$$O(a=C_j,s'=s_i,o')=\begin{cases}1-\kappa_{ij}(t_l,t_k) & o'=d\\ \\ \kappa_{ij}(t_l,t_k) & o'=\bar{d}\end{cases}, \tag{4.26}$$

where $i,j\in[1,M]$.

4.4.3.5 Belief States

In a basic POMDP model, after taking an action a and transferring to a new state s', the belief state of the new state $b(s')$ is updated based on the old belief state $b(s)$, the observation probability O, and the state transition probability P, which is represented as [100]:

$$b(s')=\frac{O(a,s',o')\sum_{s\in S}P(s'|s,a)b(s)}{\sum_{s\in S}\sum_{s'\in S}O(a,s',o')P(s'|s,a)b(s)}. \tag{4.27}$$

In an OSL problem, the belief state can be interpreted as the probability of the robot

believing that there is an odor source in a cell. Based on the defined observation probability (4.26) and the state transition probability (4.25), (4.27) can be rewritten as:

$$b(s' = s_i) = \begin{cases} \dfrac{(1 - \kappa_{ij}(t_l, t_k))b(s_i)}{\sum_{i=1}^{M}(1 - \kappa_{ij}(t_l, t_k))b(s_i)} & o' = d \\ \\ \dfrac{\kappa_{ij}(t_l, t_k)b(s_i)}{\sum_{i=1}^{M}\kappa_{ij}(t_l, t_k)b(s_i)} & o' = \bar{d} \end{cases}. \tag{4.28}$$

The above equations iteratively update belief states depending on plume detection and non-detection events. The initial belief state b_0 is defined as $1/M$ since the prior information about the odor source location is unavailable to the robot before it starts the search. However, it could be exploited through an appropriate distribution of b_0 to reflect the prior knowledge known about the source if the information regarding the source location is available prior to the search.

In summary, by calculating belief states over all states, a source probability map $\boldsymbol{b}$ that estimates the source location is obtained:

$$\boldsymbol{b} = [b(s_1), b(s_2), ..., b(s_M)]. \tag{4.29}$$

4.4.4 Plume Mapping

The plume mapping algorithm produces a plume distribution map, which indicates possible plume propagation areas. With the produced source probability map and the recorded airflow history, a HMM based plume mapping algorithm [16] is presented in this section.

Let $\alpha_j(t_0, t_k)$ denote the probability that a cell C_j contains the detectable odor plume at time t_k due to the continuous plume release by the source starting at t_0, where t_0 is the initial time that the robot records airflow measurements. Denote

$$\boldsymbol{\alpha}(t_0, t_k) = [\alpha_1(t_0, t_k), \alpha_2(t_0, t_k), ..., \alpha_M(t_0, t_k)] \tag{4.30}$$

as the vector storing this variable for each cell, which is a plume distribution map at the current time t_k.

Introduce the variable $\bar{\alpha}_j(t_0, t_k)$ representing the probability that a cell C_j contains a detectable odor plume at t_k due to a single plume release at time t_0. Define

$$\bar{\boldsymbol{\alpha}}(t_0, t_k) = [\bar{\alpha}_1(t_0, t_k), \bar{\alpha}_2(t_0, t_k), ..., \bar{\alpha}_M(t_0, t_k)] \tag{4.31}$$

as the vector form of $\bar{\alpha}_j(t_0, t_k)$. At t_0, the plume propagation has not occurred yet and plumes are at the odor source location, therefore, $\bar{\boldsymbol{\alpha}}(t_0, t_0) = \boldsymbol{b}$ since the actual odor source location is unknown. To find $\bar{\alpha}_j(t_0, t_1)$, which is the probability of a cell C_j containing plumes after one time step, the plume transitions from all other cells to the cell C_j must be considered, i.e.,

$$\bar{\alpha}_j(t_0, t_1) = \sum_{k=1}^{M} \bar{\alpha}_k(t_0, t_0) a_{kj}(t_0), \tag{4.32}$$

where $a_{kj}(t_0)$ denotes the probability of the one step transition of odor plumes from a cell C_k at time t_0 to another cell C_j at time t_1, which can be obtained from the airflow history [16]. In addition, (4.32) can be rewritten in a vector notation:

$$\bar{\boldsymbol{\alpha}}(t_0, t_1) = \bar{\boldsymbol{\alpha}}(t_0, t_0) \bar{\mathbf{A}}(t_0), \tag{4.33}$$

where $\bar{\mathbf{A}}(t_0) = [a_{kj}(t_0)] \in \mathcal{R}^{M\times M}$ is the matrix form of $a_{kj}(t_0)$. Moreover, for an arbitrary plume propagation period, i.e., $(t_k - t_0)$, $\bar{\boldsymbol{\alpha}}(t_0, t_k)$ can be calculated as:

$$\bar{\boldsymbol{\alpha}}(t_0, t_k) = \begin{cases} 0, & \text{for } t_k < t_0 \\ \boldsymbol{b}\mathbf{I}, & \text{for } t_k = t_0 \\ \boldsymbol{b}\Phi(t_0, t_k), & \text{for } t_k > t_0 \end{cases}. \tag{4.34}$$

where $\mathbf{I}$ is the identity matrix with the size of $M \times M$ and $\Phi(t_0, t_{k+1}) = \prod_{q=0}^{k} \bar{\mathbf{A}}(t_q)$.

For the continuous plume release scenario, $\boldsymbol{\alpha}(t_0, t_k)$ can be derived from the single release case $\bar{\boldsymbol{\alpha}}(t_0, t_k)$ by considering all release times from t_0 to t_k:

$$\boldsymbol{\alpha}(t_0, t_k) = \frac{1}{k+1} \sum_{q=0}^{k} \bar{\boldsymbol{\alpha}}(t_q, t_k), \tag{4.35}$$

where $1/(k+1)$ is the normalization factor to maintain $||\boldsymbol{\alpha}(t_0, t_k)||_1 = 1$. With (4.34), (4.35) can be further reduced as:

$$\begin{aligned} \boldsymbol{\alpha}(t_0, t_k) &= \frac{1}{k+1} \left[\sum_{q=0}^{k} \boldsymbol{b}\Phi(t_q, t_k) \right] \\ &= \frac{\boldsymbol{b}}{k+1} \left[\Phi(t_k, t_k) + \sum_{q=0}^{k-1} \Phi(t_q, t_k) \right] \\ &= \frac{\boldsymbol{b}}{k+1} \left[\mathbf{I} + \sum_{q=0}^{k-1} \Phi(t_q, t_k) \right] \\ &= \boldsymbol{b}\Psi(t_0, t_k), \end{aligned} \tag{4.36}$$

where $\Psi(t_0, t_k) = 1/(k+1) \left[\mathbf{I} + \sum_{q=0}^{k-1} \Phi(t_q, t_k) \right]$, which can be iteratively updated as:

$$\Psi(t_0, t_k) = \frac{1}{k+1} \left[\mathbf{I} + k\Psi(t_0, t_{k-1})\bar{\mathbf{A}}(t_{k-1}) \right]. \tag{4.37}$$

Note that, since $\bar{\mathbf{A}}(t_{k-1})$ relates to the latest airflow measurement, (4.37) encapsulates the airflow history over the entire search time (i.e., from t_0 to t_k).

In summary, when the source probability map $\boldsymbol{b}$ is available at the current time t_k, a plume propagation map $\boldsymbol{\alpha}(t_0, t_k)$ can be obtained by (4.36), and if $\boldsymbol{b}$ is updated in the next time step, by updating Ψ with the latest airflow measurements, a renewed plume distribution map based on the new source probability map can be obtained.

4.5 Planning

4.5.1 Generate Reward Functions with Fuzzy Inference

After the source probability map $\boldsymbol{b}$ and the plume distribution map $\boldsymbol{\alpha}(t_0, t_k)$ are obtained, information from two maps is fused and assigned to reward functions. The information provided by two maps is complementary for determining robot behaviors, i.e., the robot either moves to the estimated source location or to the possible plume areas. Thus, a weighted superposition pattern is adopted to combine two maps, and values from two maps are normalized with the min-max normalization before the combination.

Let define four constants, b_{max}, b_{min}, α_{max}, and α_{min}, as maximal and minimal values in $\boldsymbol{b}$ and $\boldsymbol{\alpha}(t_0, t_l)$, respectively. These constants can be determined before computing reward functions. For an action that moves the robot into a cell C_j, the reward function is:

$$\begin{aligned} &R(s, a = C_j) \\ &= \lambda \frac{b(s_j) - b_{min}}{b_{max} - b_{min}} + (1 - \lambda) \frac{\alpha_j(t_0, t_k) - \alpha_{min}}{\alpha_{max} - \alpha_{min}}, \end{aligned} \tag{4.38}$$

where $\lambda \in [0, 1]$ is the fusion coefficient that controls the balance of two maps. When $\lambda >$ 0.5, the source probability map is chiefly dominated in reward functions, which results

the robot surging to the estimated source location, i.e., the exploitation. Conversely, when $\lambda < 0.5$, the robot moves to possible plume areas since the plume distribution map outweighs the counterpart in reward functions, i.e., the exploration. An ideal value of λ should be adaptive for different search circumstances to generate the optimal search objective.

The primary hurdle of determining the value of λ is the vagueness of search circumstances. A critical question to ask is that under what conditions the robot should choose its search objective as the exploration or the exploitation. Attempts that use mathematical methods to quantitatively analyze differences of search conditions and assign a precise λ for different search circumstances are hard to implement due to uncertainties of the source location and the search environment.

Inspired by implementations of fuzzy theory in the field of decision making [83] and data classification [84], which successfully handles the problems with vagueness and uncertainties, a fuzzy inference approach is employed. In fuzzy theory, vague variables and environments can be handled in a deterministic manner via linguistic descriptions and rules. By analyzing sensor measurements, such as plume concentrations, search circumstances are expected to be identified. Then, the corresponding λ that dynamically balances the exploitation and exploration of the odor source information can be generated based on defined fuzzy rules.

As shown in Fig. 4.7, procedures of the proposed fuzzy controller include fuzzification, defining fuzzy rules, and defuzzification [101].

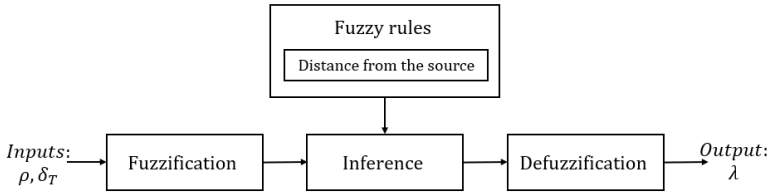


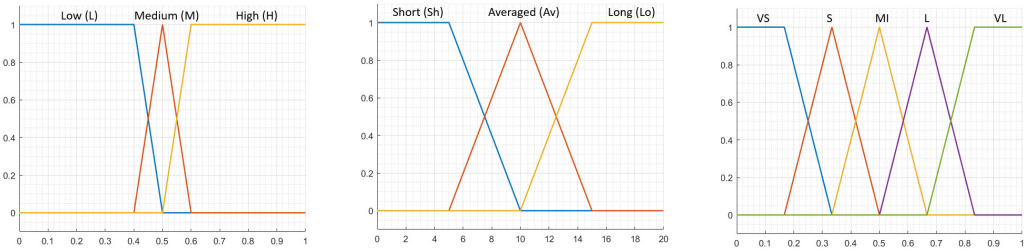
FIGURE 4.7: Structure of the proposed fuzzy controller. ρ and δ_T are sensed plume concentration and plume non-detection period respectively, and λ is the fusion coefficient.

4.5.1.1 Fuzzification

Fuzzification is the process that changes real scalar values of antecedent and consequent (i.e., inputs and outputs) into fuzzy values, which are the degree of uncertainty that scalar values belong in a fuzzy set.

In this work, inputs are utilized to conjecture the distance from the robot to the odor source and the output is the value of λ. If the robot is close to the source, it surges toward the source ($\lambda > 0.5$); otherwise, it leans to detect plumes to gather more information ($\lambda < 0.5$). The sensed plume concentration at the robot location $\rho(\mathbf{X}_v(t), t)$ (for the simplification purpose, we use ρ as plume concentration measurements in the rest of paper) is set as an input, and due to the existence of local concentration maxima along the plume trajectory [87], the plume non-detection period δ_T, i.e., the time interval between two detection events, is added to inputs. Since the positions of local concentration maxima are time varying in a turbulent flow environment [88], if the robot senses consecutive high odor concentrations in a short period, it is very likely that the robot is near to the source.

Fig. 4.8 shows plots of membership functions of inputs and the output, which are determined based on the distribution of measured data from experiments. As shown in Fig. 4.8, all membership functions are triangular. Three fuzzy sets have been defined to cover the discourse of universe of the sensed plume concentration ρ, namely Low



(a) (b) (c)

FIGURE 4.8: Fuzzy sets and membership functions of antecedents and consequent. (a) Sensed plume concentration, ρ. (b) Plume non-detection period, δ_T. (c) Fusion coefficient, λ.

(L), Medium (M), and High (H). The discourse of universe of the plume non-detection period δ_T is also covered by three fuzzy sets, namely Short (Sh), Averaged (Av), and Long (Lo). For the output λ, five fuzzy sets, namely Very Small (VS), Small (S), Middle (MI), Large (L), and Very Large (VL), are defined to cover its discourse of universe.

4.5.1.2 Fuzzy Rules

Fuzzy rules in the fuzzy inference theory are presented in a 'IF-THEN' format, which determine search strategies of the robot. In this work, fuzzy rules are designed based on moth odor searching behaviors [26]. As mentioned, previous researchers [33, 37, 38] have summarized these behaviors into a 'surge/casting' model and demonstrated the validity of implementing this model on robots in OSL problems. Borrowing this idea, we want the robot to explore if plumes are absent (like the moth's casting behavior) and to exploit when the robot is in plumes (like the moth's surge behavior). To achieve this mechanism, the distance from the robot to the odor source is estimated and monitored: if the robot is far from the source, the robot inclines to find plumes, i.e., exploration; otherwise, the robot tends to search the source, i.e., exploitation.

The inclination of changing λ is that: when ρ is high and δ_T is short, the robot is very likely being close to the odor source; thus, λ is large. On the other hand, when ρ is

TABLE 4.1: List of Fuzzy Rules

Rule No.	Rule
$\mathcal{F}^1$	IF ρ is L AND δ_T is Sh, THEN λ is MI
$\mathcal{F}^2$	IF ρ is L AND δ_T is Av, THEN λ is S
$\mathcal{F}^3$	IF ρ is L AND δ_T is Lo, THEN λ is VS
$\mathcal{F}^4$	IF ρ is M AND δ_T is Sh, THEN λ is L
$\mathcal{F}^5$	IF ρ is M AND δ_T is Av, THEN λ is MI
$\mathcal{F}^6$	IF ρ is M AND δ_T is Lo, THEN λ is S
$\mathcal{F}^7$	IF ρ is H AND δ_T is Sh, THEN λ is VL
$\mathcal{F}^8$	IF ρ is H AND δ_T is Av, THEN λ is VL
$\mathcal{F}^9$	IF ρ is H AND δ_T is Lo, THEN λ is L

low and δ_T is long, the robot is probably far from the source; thus, λ is small. In a 'IF-THEN' format, the above rules are represented as:

$\mathcal{F}^1 = \{$IF ρ is H AND δ_T is Sh, THEN λ is VL $\}$,

$\mathcal{F}^2 = \{$IF ρ is L AND δ_T is Lo, THEN λ is VS $\}$.

Enumerate all possible combinations of antecedents and the corresponding consequent, a rule table (Table 4.1) can be obtained.

4.5.1.3 Defuzzification

The centroid method [89] is chosen as the defuzzification algorithm, which can be expressed as:

$$U_0 = \frac{\sum_{i=1}^{n} U_i \cdot \mu\left(U_i\right)}{\sum_{i=1}^{n} \mu\left(U_i\right)}, \tag{4.39}$$

where U_0 is the output (i.e., the value of λ), i is the index of rules $i \in [1, 9]$, $\mu(U_i)$ is the truth value of result membership function for the ith rule, and U_i is the value where the result membership function is maximum over the output variable fuzzy set range.

Fig. 4.9 presents the result of the proposed fuzzy controller. In general, a small ρ and a long δ_T produce a trivial λ value that emphasises the plume mapping information in reward functions, and the opposite combination of ρ and δ_T (i.e., a large ρ and a

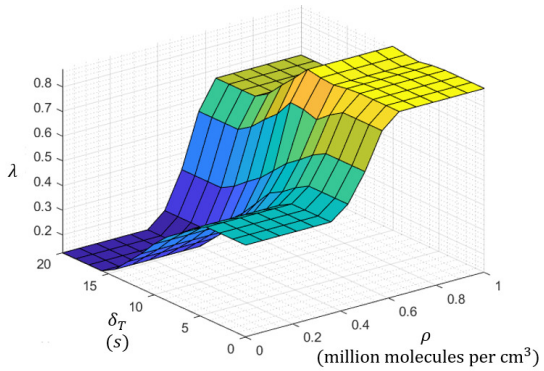


FIGURE 4.9: The result of the proposed fuzzy controller. In the plot, the horizontal axes are two inputs, the sensed odor concentration ρ and the plume non-detection period δ_T, and the vertical axis is the output, the fusion coefficient λ.

short δ_T) provides a large λ that prioritizes the source mapping information in reward functions.

4.5.2 Solve for the Optimal Policy

After reward functions are determined, a search route is generated in the planning procedure. Given the current reward functions, we adopt a value iteration method (Algorithm 4) to fast determine the optimal policy, i.e., the search route. The motivation of using this method is to reduce the processing time, which allows the robot to timely respond to new plume observations. By contrast, solving the POMDP [102] is also feasible to obtain the search route, but, considering the large size of the hidden state space defined in our POMDP, this approach becomes time-consuming and intractable. The ability of fast solving the searching path is one of our main concerns since the ultimate goal of this work is implementing this algorithm on a mobile robot, which has limited onboard computational resources.

As shown in Algorithm 4, value functions of all cells are initialized as 0. If the robot is currently located in a cell C_i, it could choose one of eight actions (see Fig. 4.6) and enter the corresponding cell. Note that, based on the reliable maneuverability of the

Algorithm 4 Value Iteration Based Planning Algorithm

Initialize Value Functions $V(C_i) = 0$, $i \in [1, M]$
Calculate Reward Functions $R(s, a)$ for all cells based on (4.38)
Set the convergence tolerance ϵ
while $\Delta \geq \epsilon$ **do**
    $\delta = 0$
    for $i \in [1, M]$ **do**
        $v = V(C_i)$
        $V(C_i) = \mathbf{max}_{a \in \mathcal{A}} \left(R(s, a) + \gamma V(a) \right)$
        $\Delta = \mathbf{max}(\Delta, |v - V(C_i)|)$
    end for
end while
Generate the optimal policy $\pi^* = \mathbf{argmax} V(C_i)$

ground robot, it is assumed that the transition of robot positions is deterministic, i.e., after taking an action, the robot can correctly enter the corresponding cell. Based on the Bellman optimality equation [97], the value function of a cell C_i can be calculated by:

$$V(C_i) = \max_{a \in \mathcal{A}} \left(R(s, a) + \gamma V\left(a\right) \right). \tag{4.40}$$

where $\gamma \in [0, 1]$ is the discounting factor that penalizes future rewards. In this work, the larger the γ is, the broader region the robot considers in planning the search route. In implementations, we set γ to be 0.9.

A convergence tolerance ϵ is defined to check whether or not value functions converge. The maximal update of value functions (i.e., Δ in Algorithm 4) is compared with ϵ, and value functions are considered as converged if $\Delta < \epsilon$, otherwise value functions keep updating. To balance the trade-off between the navigation performance and the processing time, we set ϵ as 10^{-6} in experiments to obtain a well algorithm performance and to save the processing time. After value functions converge, the optimal policy is obtained by selecting the optimal action with the maximal value function. Thus, a series of optimal actions can be obtained between the robot current position and the maximal reward location, which is a search route that guides the robot toward the position with

the most odor source information.

It should be noted that this optimal policy is obtained based on the current reward functions and is not permanent. In every time step, new observations, i.e., plume detection and non-detection events, update reward functions via (4.38), and a new optimal policy will be determined via Algorithm 4. The new policy overwrites the old one, which allows the robot to timely adjust its searching targets (exploration or exploitation) and trajectories to intelligently fit new observations. In general, the overall searching trajectory is a combination of a sequence of optimal policies generated from different reward functions.

4.6 Experiments

4.6.1 Experiment Designs

To evaluate the performance of the proposed navigation method, around 60 tests have been conducted on the simulation program. The simulation program and the implemented mobile robot are introduced in 3.4.1.

These tests can be separated into three groups. In the first group of tests, the proposed source mapping, plume mapping, and fusion algorithms are implemented in a laminar flow environment to verify their validities. Source and plume estimations are presented and compared with actual source and plume locations. Tests in the second group are carried out to investigate the effectiveness of implementing the proposed navigation method in a turbulent flow environment. Results are compared with a moth-inspired method [33] and a Bayesian inference method [51], which are typical plume tracing approaches in categories of bio-inspired and engineering-based methods, respectively. In

the last group of tests, the robustness of the proposed navigation method is evaluated. Various search conditions, including varying initial search positions and airflow fields, are designed. Similar to the second group of tests, implementation results of the proposed navigation method are compared with two typical plume tracing methods.

4.6.2 Group 1: Implementation in a Laminar Flow Environment

This section demonstrates the results of implementing the proposed source mapping (Section 4.4.3.5), plume mapping (Section 4.4.4), and fusion algorithms (Section 4.5) in the simulation. The robot starts the OSL task at $(60, -40)$ m (or in the cell C_{64}) in a laminar flow environment, where $\mathbf{U}_0 = (2, 0)$ m/s and $\varsigma = 5$.

Fig. 4.10 shows results of the above algorithms after the robot detecting odor plumes for the first time. To visualize algorithm results, cells in the search area are painted with various colors, where darker cells have the higher values (red: largest, white: lowest). Depending on the implementing algorithm, the value of a cell could be the probability of containing the source (i.e., the result of source mapping), the probability of carrying plumes (i.e., the result of plume mapping), and the reward function (i.e., the result of fusion algorithm).

For the source mapping algorithm, it can be observed in Fig. 4.10(a) that possible source locations are narrowed to upflow areas of the plume detection location, and as shown in Fig. 4.10(d), the cell with the maximal probability of containing the odor source is close to the actual odor source location. For the plume mapping algorithm, Fig. 4.10(b) and 4.10(e) illustrate plume estimations and the plume distribution map, respectively. As shown in these two diagrams, plume estimations correctly overlap with the actual

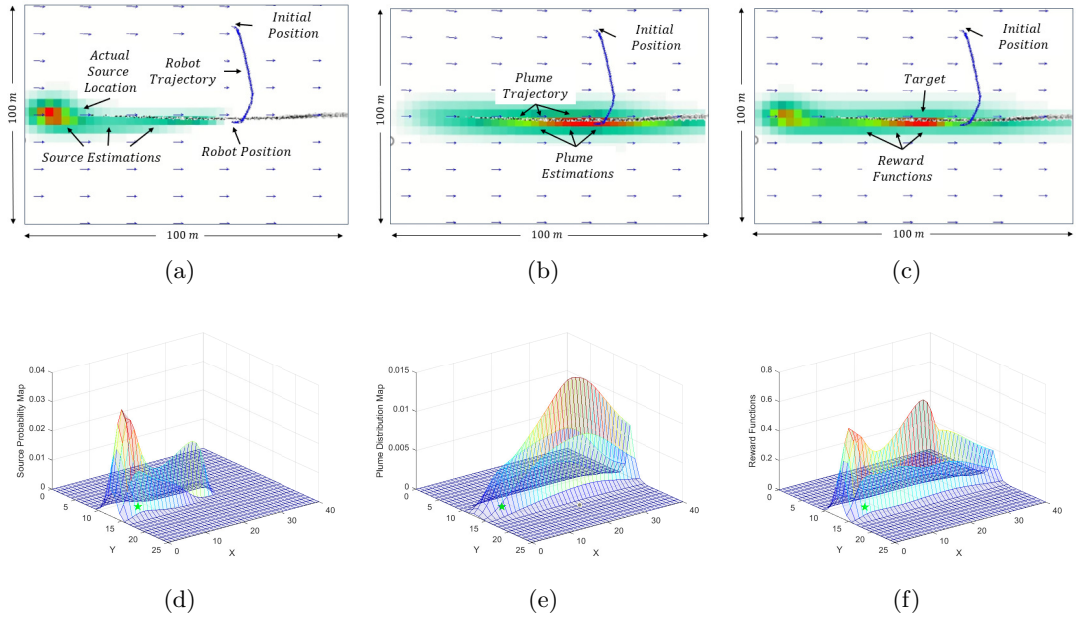


FIGURE 4.10: Results of the source mapping, plume mapping, and fusion algorithms after the robot detecting plumes for the first time. In the top row of diagrams (i.e., (a), (b), and (c)), cells in the search area are painted with different colors to reflect various values of algorithm results. In the bottom row of diagrams (i.e., (d), (e), and (f)), the horizontal plane is the grid that covers the search area, and the star mark in the horizontal plane indicates the actual odor source location. (a) Source estimations. (b) Plume estimations. (c) Reward functions. (d) Source probability map. (e) Plume distribution map. (f) The plot of reward functions.

plume trajectory, and the cell with the peak probability of containing plumes is located at the upflow area, which is a reasonable estimation of plume propagation areas.

For the fusion algorithm, due to the low sensed plume concentration ρ and the short plume non-detection period δ_T, the fusion coefficient λ is middle according to the defined fuzzy rules. Fig. 4.10(c) presents the distribution of reward functions over the search area. It can be seen that reward functions cover the plume trajectory, and the maximal reward location (i.e., the target of the path planning algorithm), as shown in Fig. 4.10(f), is at the upflow area of the last plume detection location. Test results reveal that reward functions are instructive for the robot to collect more odor source information since the robot is expected to detect more plumes at the maximal reward location.

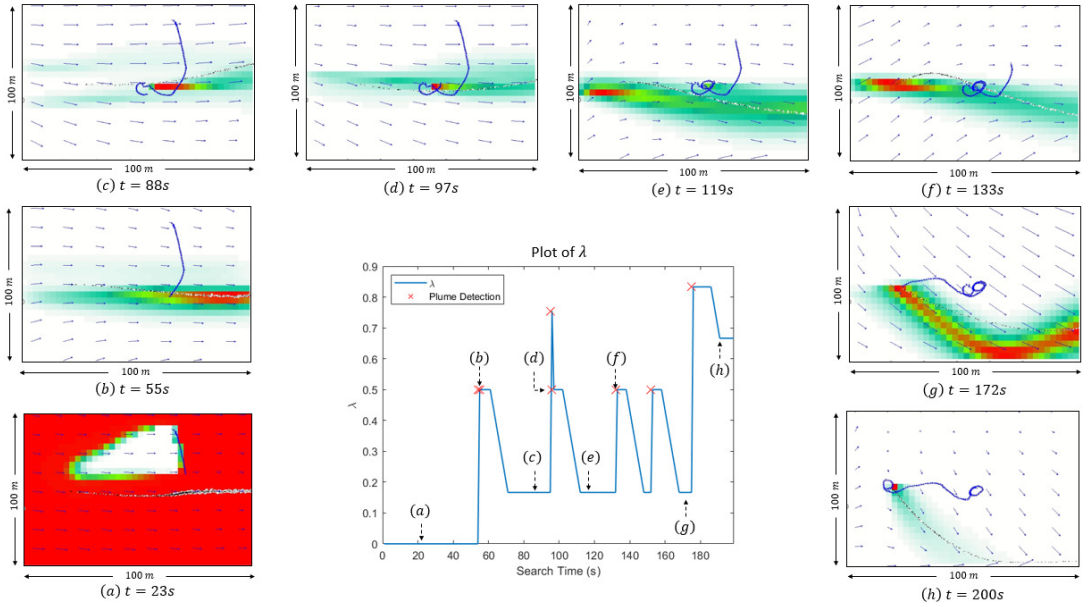


FIGURE 4.11: Robot search trajectories and reward functions at different time steps in an environment with turbulent flows. The plot of the fusion coefficient λ versus the search time t is presented at the center of the diagram, where cross marks indicates plume detection events. Diagrams around the center plot are robot trajectories and reward functions at different time steps. For each of these diagrams, the robot trajectory is represented by the trail of dark arrows; the grey-scale patchy trail in the middle of the background indicates the simulated plume trajectory; cells are painted with colors according to their reward values, where darker cells have higher reward values (red: largest, white: smallest).

4.6.3 Group 2: Implementation in a Turbulent Flow Environment

In this test, the proposed navigation method is implemented in an environment with turbulent flows ($\mathbf{U}_0 = (3, 0.5)$ m/s and $\varsigma = 30$). Fig. 4.11 shows reward functions over the search area at different times in an OSL task and the plot of the fusion coefficient λ. Similar to the first group of tests, cells in the search area are painted with different colors, where the darker color indicates the higher value in reward functions and vice versa.

The robot starts at the cell C_{64} and adopts a 'zigzag' search trajectory at the beginning of the OSL task, where the source mapping algorithm is activated simultaneously. As shown in Fig. 4.11(a), the source mapping algorithm excludes possible source locations from the upflow areas of the robot (white areas) since the robot does not detect plumes.

At 55 s, the robot detects plumes for the first time, and the plume mapping and fusion algorithms are activated to generate reward functions.

From 55 s to 172 s, the robot is in the exploration, where the robot is encouraged to detect plumes and gather odor source information. Specifically, it can be observed in the plot of λ that the value of λ fluctuates between 0 and 0.5 due to the low sensed odor concentration ρ and the long plume non-detection period δ_T. As the result, plume estimations outweigh source estimations in reward functions, which drives the robot to seek plumes. Note that, at 96 s and 97 s, λ rises to 0.75 but quickly falls to 0.5. It is because the robot senses a local concentration maximum at 96 s, but the successive sensed concentration is not as high as the previous, which contributes the drop of λ. After successively detecting high concentration plumes at 175 s, the robot is in the exploitation state. At 175 s, λ rises to 0.83, which indicates that the robot is near the source, and the robot surges toward the estimated source location. At 200 s, reward functions converge to a single cell C_{292} (i.e., the red cell in Fig. 4.11(h)), which overlaps the actual odor source location, and the robot successfully finds the odor source location.

A moth-inspired method [33] and a Bayesian inference method [51] are implemented in the same environment to compare with the proposed method. Fig. 4.12 shows search trajectories, and Table. 4.2 compares search times and travel distances of three navigation methods. As shown in Fig. 4.12(a), the moth-inspired method fails to find the odor source within the time limit (500 s). The primary reason is that the plume trajectory alters rapidly in a turbulent flow environment, thus, the robot can barely stay in plumes and surge upflow to seek the odor source. For the Bayesian inference method, it can be observed in Fig. 4.12(b) that due to the lack of plume estimations, the robot constantly circulates and tries to detect new plumes after the second plume detection event (i.e., the middle cross on the searching trajectory). As the result, the

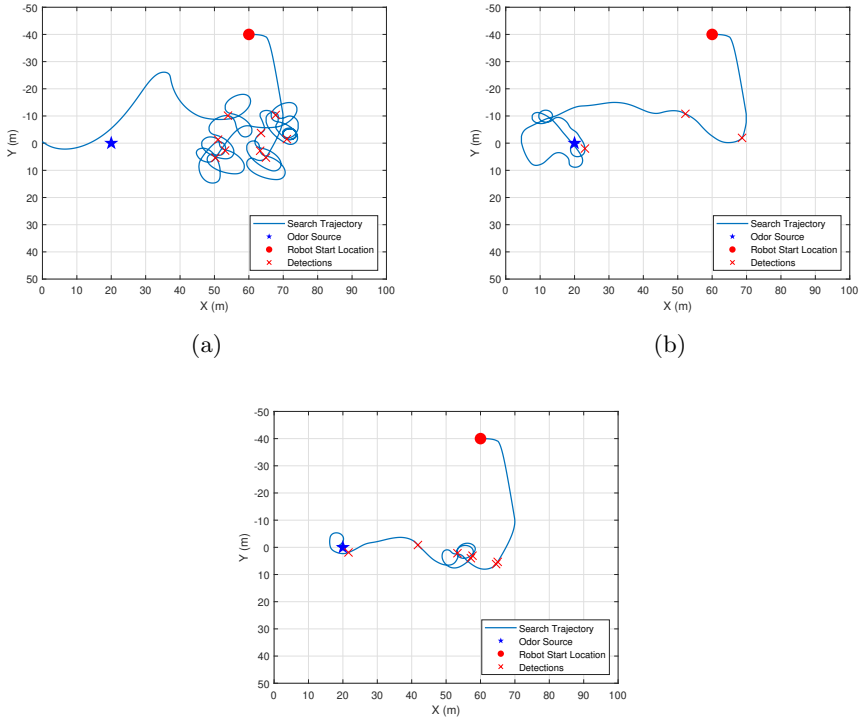


FIGURE 4.12: Search trajectories of three navigation methods in a turbulent flow environment. (a) Moth-inspired method. (b) Bayesian-based method. (c) Proposed method.

TABLE 4.2: Search Time and Travel Distance of Three Navigation Methods in a Turbulent Flow Environment

	Moth-inspired method	Bayesian inference method	Proposed method
Search time (s)	- [1]	251	200
Travel distance (m)	-	223.96	172.07

[1] -: Fail to locate the source within 500 s.

robot spends a lot of time to recover from plume non-detection events. By contrast, the proposed navigation method achieves the best performance with the shortest search time and travel distance. Compared to the search trajectory of the Bayesian inference method, the robot implemented with the proposed navigation method detects more plumes, and after several plume detection events, the robot surges toward estimated source location and correctly finds the odor source eventually. Results in this test verify the validity of implementing the proposed method in a turbulent flow environment.

4.6.4 Group 3: Varying Search Conditions

In this test, the robustness of the proposed navigation method in varying search conditions is investigated. Two scenarios have been designed in this test, which are listed in the following:

- Scenario 1: the robot starts an OSL task at different initial positions in a turbulent flow environment ($\mathbf{U}_0 = (2, 0)$ m/s and $\varsigma = 15$).
- Scenario 2: the robot starts at the same initial position, but airflow fields are varying ($u_x \in [1, 3]$ m/s, $u_y \in [-0.5, 0.5]$ m/s, and $\varsigma \in [10, 30]$).

4.6.4.1 Results of Scenario 1

Six tests have been conducted in Scenario 1, and Fig. 4.13 presents search trajectories of all tests. It can be observed in Fig. 4.13 that all search trajectories terminates at the actual source location, which is marked by a star, i.e., the robot correctly finds the odor source in all tests, which demonstrates the validity of the proposed navigation method with varying initial searching positions.

4.6.4.2 Results of Scenario 2

In order to evaluate the performance of the proposed navigation method, the moth-inspired method and the Bayesian-based method are also implemented and compared in this test. For each navigation method, the test is repeated 15 times in environments with varying airflow conditions. Table 4.3 presents airflow conditions of 15 tests and search times of three navigation methods in the corresponding environment, and Table

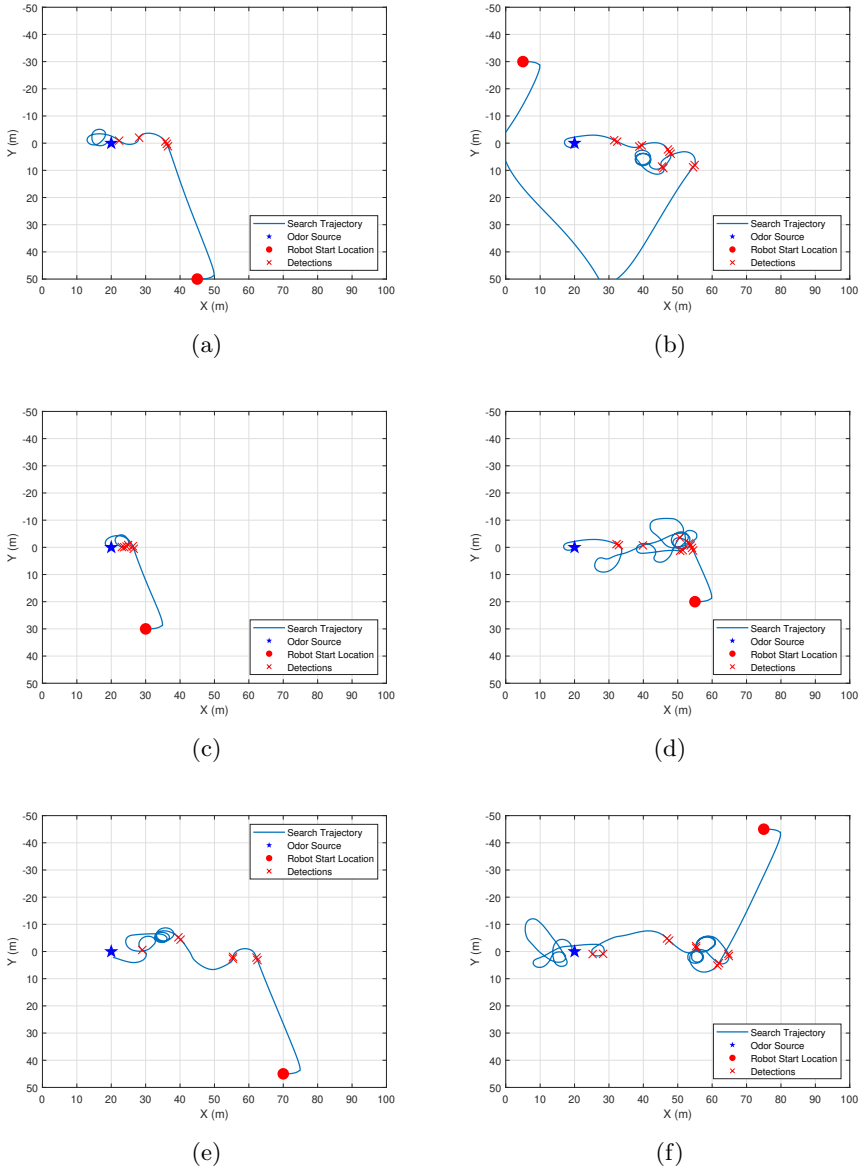


FIGURE 4.13: Search trajectories of the proposed navigation method at different initial positions. The robot starts an OSL task from (a) $(45, 50)$ m; (b) $(5, -30)$ m; (c) $(30, 30)$ m; (d) $(55, 20)$ m; (e) $(70, 45)$ m; (f) $(70, -45)$ m.

4.4 lists statistical results of all tests. It can be seen from Table 4.3 that the moth-inspired method barely succeeds to find the odor source in turbulent flow environments. Comparing to the Bayesian inference method, the robot with the proposed method achieves a higher success rate (100% vs 80%) and a shorter averaged search time (199.5 s vs 299.1 s), which illustrates the effectiveness of the proposed method in varying airflow environments.

TABLE 4.3: Environmental Settings and Search Times of Different Navigation Methods

Tests	Mean Wind Velocity, $\mathbf{U}_0$ (m/s)	Gaussian Noise Variance, ς	Moth-inspired Method (s)	Bayesian inference Method (s)	Proposed Method (s)
Test 1	[1, -0.5]	10	-	245	171
Test 2	[1.2, 0.5]	11	-	160	177
Test 3	[2, 0]	15	-	223	150
Test 4	[2, 0.3]	18	275	367	229
Test 5	[3, 0.5]	12	-	489	214
Test 6	[3, -0.3]	20	-	418	194
Test 7	[2, 0.4]	30	-	-	282
Test 8	[3, -0.5]	25	-	248	152
Test 9	[2, -0.2]	24	-	303	231
Test 10	[1, 0.2]	28	-	-	152
Test 11	[2.4, 0.2]	28	-	287	283
Test 12	[2.1, 0.5]	13	-	243	224
Test 13	[1.2, -0.3]	12	-	-	213
Test 14	[1.1, -0.1]	28	-	375	194
Test 15	[2, -0.4]	10	-	231	126

-: Fail to locate the source within 500 s.

TABLE 4.4: Statistical Results of Repeated Tests and the Comparison of Three Navigation Methods

	Total Test	Successful Test	Success Rate	Averaged Time (s)
Moth-inspired Method	15	1	6.7%	275
Bayesian-based Method	15	12	80%	299.1
Proposed Method	15	15	100%	199.5

4.6.5 Discussion

Results in the above tests reveal the effectiveness of the proposed olfactory-based navigation method. However, it is worth mentioning that the limitation of our method is the lack of global wind measurements. Due to this reason, as mentioned in Section 4.4.2.2, wind measurements at robot positions are utilized to estimate plume advection distances. This approximation introduces additional errors if the robot is in a highly turbulent flow environment. i.e., wind directions and velocities have a huge variance in space. It can be observed in Table 4.3 that the searching time grows significantly if wind fields are highly turbulent (e.g., Test 4, 7, and 11). This issue could be alleviated with

the multi-agent searching algorithm. By employing multiple robots, wind information at different locations are obtained, and a comprehensive wind map over the searching area could be derived. The design and implementation of the multi-agent searching algorithm is one of our prospective research directions.

4.7 Summary of this Chapter

This chapter presents an olfactory-based navigation algorithm based on model-based RL and fuzzy inference methods. The OSL problem is modeled as a model-based RL problem, in which a POMDP is used to model the plume tracing procedure. By doing this, a source probability map and a plume distribution map are generated. The information from both maps is fused by a fuzzy inference based fuzzy controller and assigned to reward functions, and the value iteration method is adopted to solve for the optimal policy. Experiment results show that the proposed navigation method is valid in turbulent flow environments. Besides, comparing to the moth-inspired method and the Bayesian-based method, the proposed method is more effective and intelligent in turbulent flow environments.

Chapter 5

Olfactory-Based Navigation via Machine Learning and Artificial Intelligence Methods

This chapter presents machine learning (ML) and artificial intelligence (AI) based plume tracing algorithms for using on a mobile robot to find odor sources in unknown environments. The core idea of this type of algorithms is to train an intelligent ML model that controls the plume tracing robot to effectively and efficiently find an odor source based on robot perceptions. The training data can be acquired by implementing other successful plume tracing algorithms in OSL tests, where sensor readings and robot commands are recorded. In this chapter, three ML models are employed, including an adaptive neuro-fuzzy inference system (ANFIS), feedforward neural network (FNN), and convolutional neural network (CNN), to navigate the robot in finding the odor source in turbulent flow environments. Two traditional olfactory-based navigation algorithms, including a bio-inspired method and an engineering-based method, were implemented in simulated

OSL trials to generate training data. After the supervised training, proposed ML models were validated in the simulated OSL environment.

5.1 Motivation and Research Niche

In previous studies, it was found that the limitation of bio-inspired methods is the lack of odor plume estimations. Thus, when odor plumes are absent, the robot can only perform a time-consuming 'casting' behavior to find plumes. What is worse is that this method usually fails in a turbulent flow environment since odor plume trajectories change rapidly in this scenario. As for an engineering-based method, the requirement for the high computational capacity to online estimate odor source locations restrains its applications when the search area is large and complex. A desired olfactory-based navigation algorithm should be light-weighted, i.e., it does not have a high computational demand, while efficient and capable enough to locate odor sources in different flow conditions.

Motivated by this consideration, the author leverages the OSL problem via ML and AI algorithms. The principal idea of ML&AI technologies is letting the computer recognize the pattern of performing a task by relying on data sets without providing explicit instructions [103]. A wide spectrum of ML-related applications [104–106] has demonstrated the powerful ability of ML algorithms to estimate uncertainties, such as feature extraction [104], medical diagnosis [105], and self-driving vehicles [106]. The motivation of applying ML algorithms to solve an OSL problem is that we want an ML model to learn an odor source searching strategy from benefits of other successful olfactory-based

navigation methods without explicating the specific searching algorithms and rules. Besides, an ML algorithm is light-weighted, which is suitable for working with autonomous vehicles.

There are many benefits for applying ML methods on the OSL problem: 1), compared to complex engineering-based navigation methods, the query time of ML models is predictable and unaffected by search environments, which is suitable for implementing on mobile robots; 2) ML models can learn other successful navigation methods from demonstrations without explicating the specific searching algorithm; 3) ML models can continually improve the searching performance by adding more search examples in training data sets. However, it should be mentioned that the challenging part of this application is to obtain training data sets since OSL experiments are expensive to repeatedly perform in different airflow conditions.

In this chapter, multiple ML models are adapted to solve the OSL problem, including the model of an adaptive neuro-fuzzy inference system [107] and two types of deep neural networks (DNNs) [103], namely FNN and CNN. During the plume tracing process, the proposed ML models produce suitable robot commands based on onboard sensor measurements. Two paradigms from categories of bio-inspired and engineering-based methods, namely moth-inspired [33] and Bayesian-inference [51] methods, are employed as expert methods to generate training data sets in a realistic simulation program. A simulation program that emulates time-varying airflow fields and corresponding plume trajectories is selected as the platform to generated training data and validate the trained ML model. Considering the difficulty of repeatedly conducting the actual OSL experiments and the requirement of the large quantity training data, collecting training data from a realistic simulation program is an acceptable option.

After the supervised training [108], the proposed ML models are implemented in OSL tests with various search conditions, where the success rate and the averaged search time are calculated. To analyze the generalization error of the trained ML models, OSL tests are conducted in previously unseen environments. Additionally, trained ML models are also compared with traditional navigation methods to evaluate the validity of the proposed methods.

5.2 Prepare Training Data Sets

5.2.1 The Simulated Searching Environment

As mentioned, the OSL is considered as a two-dimensional (2-D) problem since the aimed implementation robotic platform is a ground mobile robot. A realistic OSL simulation program [8] that emulates filament-based odor plume trajectories in time-varying airflow fields is employed as the platform to generate training data sets. Fig. 5.1(a) shows the simulated search area, where the size is 100×100 m^2. Over the search area, a coordinate system $(x - y)$ is constructed, and an odor source is located at $(20, 0)$ m and releases 10 plumes per second. Released plumes form a circular plume trajectory as plotted by a grey-scale patchy trail. The shape and position of the plume trajectory are varying with local winds, which are indicated by arrows in the background. Wind vectors are calculated from time-varying boundary conditions that are generated by a mean flow ($\mathbf{U}_0 = (u_{0x}, u_{0y})$) plus Gaussian white noises (0 mean and ς variance). Varying airflow fields and the corresponding odor plume trajectories can be obtained by adjusting values of $\mathbf{U}_0$ and ς.

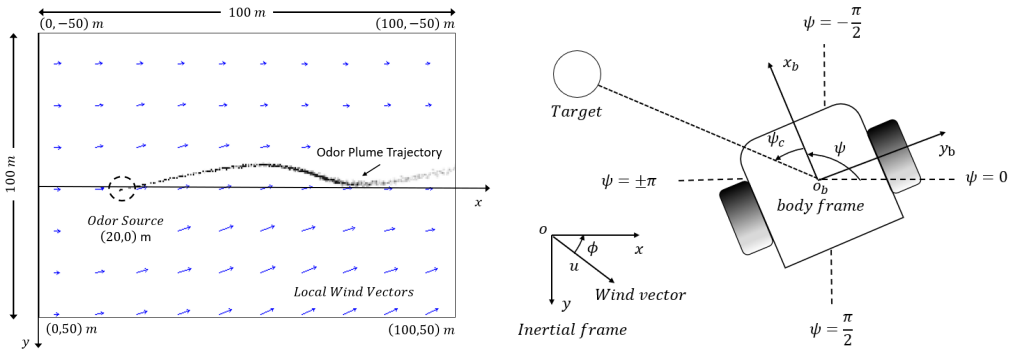


FIGURE 5.1: (a)The simulated search area. (b)The two-wheeled mobile robot used in the simulation program. Wind velocity u, wind direction ϕ, and robot positions (x, y) are measured from the onboard anemometer and the positioning sensor in the inertial frame. The robot heading ψ is monitored by the onboard compass. The heading command ψ_c is defined as the difference between the current heading and the target direction.

In the simulation program, a two-wheeled mobile robot, as shown in Fig. 5.1(b), is employed as the platform to implement the ANFIS model after the training. It is assumed that the robot is equipped with a chemical sensor, an anemometer, a positioning sensor, and a compass which measure odor concentrations ρ, wind speeds u and directions ϕ in the inertial frame, the robot position (x, y) in the inertial frame, and the robot heading ψ respectively.

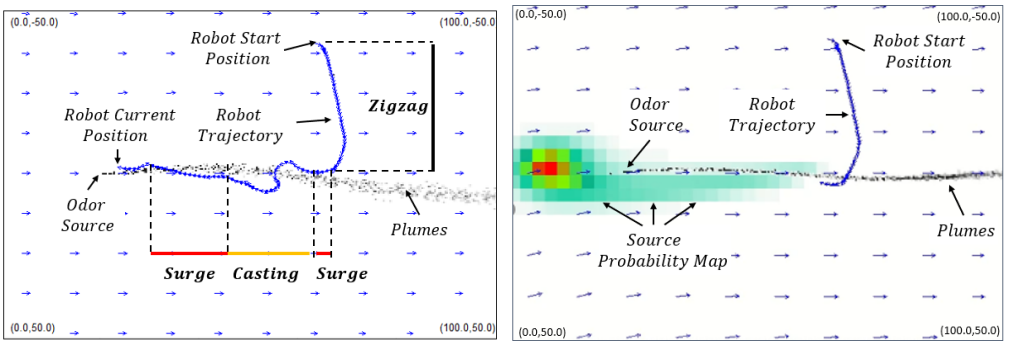
To control the mobile robot in a 2-D plane, only speed and heading commands are needed. To simplify the problem, the robot moves in a constant speed, i.e., 1 m/s, and the heading commands ψ_c are generated from the implementing olfactory-based navigation method. It should be mentioned that the range of heading commands is from $-\pi$ to π, which specifies the rotation direction (negative: anti-clockwise; positive: clockwise) and angle. Comparing to the large scale of the search area, the size of the robot is negligible. Thus, the robot is approximated as a single point in the simulation.

5.2.2 Introduction of Two 'Instructors'

Moth-inspired [33] and Bayesian-inference [51] olfactory-based navigation methods are selected as 'instructors' to generate training data sets, which are paradigms in bio-inspired and engineering-based methods, respectively.

As mentioned, the core idea of the moth-inspired method is to mimic the mate-seeking behavior of male moths, which can be summarized as a two-phase searching strategy: 'surge' and 'casting'. Specifically, the 'surge' searching phase is triggered when the robot detects odor plumes, where the robot stays inside plumes and moves upwind. When the robot is out of plumes, the 'casting' searching phase is activated, where the robot casts in circles to re-detect plumes. Fig. 5.2(a) demonstrates the robot searching trajectory generated by implementing this method. The robot first adopts a 'zigzag' searching trajectory to detect the existence of odor plumes in the search area, and after it detects plumes for the first time, the robot surges upwind until it is out of plumes. Then, the robot traverses the wind to recover plumes. The robot switches between 'surge' and 'casting' searching phases depending on whether or not the robot detecting plumes. Experiment results show that this method is valid in a laminar flow environment, where the plume trajectory is relatively stable and continuous.

For the Bayesian-inference method, it utilizes measured wind information and a Gaussian plume dispersion model to reversely deduce the odor source location. Fundamental procedures are twofold: source mapping and path planning. In the source mapping procedure, the search area is divided into multiple cells, and a Gaussian plume dispersion model is employed to calculate the probability of the robot detecting or not detecting plumes in an arbitrary cell. Then, a source probability map that estimates possible odor source locations over the search area is generated and iteratively updated with



(a) (b)

FIGURE 5.2: Searching trajectories generated by traditional olfactory-based navigation methods. (a) Moth-inspired method. Over the searching trajectory, different searching phases are highlighted with color bars. Black, red, and yellow represent 'zigzag', 'surge', and 'casting', respectively. (b) Bayesian-inference method. To visualize the source probability map, the searching area is painted with various colorful cells, where darker cells have higher probability of containing the odor source, and the lighter cells have less (red: highest, white: lowest).

plume detection and non-detection events. In the path planning procedure, the robot first adopts a 'zigzag' search trajectory to detect the existence of odor plumes. After it senses odor plumes for the first time, a possible odor source location indicated by the source probability map, i.e., the cell with the highest probability of containing the odor source, is obtained. Then, a path planner is employed to guide the robot moving to the source estimations. Fig. 5.2(b) shows a snapshot of the source probability map that is generated after the robot detects odor plumes for the first time. Comparing to the moth-inspired method, experiment results indicate that the Bayesian-inference method carries a higher success rate of locating the odor source in a turbulent flow environment.

Comparing these two methods, the moth-inspired method is more effective in a laminar flow environment, where the plume trajectory is stable and continuous, but search efficiency of this method is not ideal if the odor source is located in a turbulent flow environment since it is hard for the robot to stay inside a rapidly changing plume trajectory. As for the Bayesian-inference method, by recording airflow information, it can accurately estimate source locations. In turbulent flow environments, the performance

of this method is much better than the moth-inspired approach. However, it requires high computational capacity to build the source probability map over the search area if the search area is large.

5.3 Train an ML Model with ANFIS

5.3.1 ANFIS Basics

Jang [107] proposed the structure of the ANFIS, which is a feed-forward network that combines neural networks and fuzzy logic theory to map the relationships between inputs and outputs. The innovation of this method is that it does not require expert knowledge to assign parameters of a fuzzy inference system, but utilizes neural network learning algorithms to tune parameters.

Fig. 5.3 shows a five-layer ANFIS architecture with two inputs (x and y) and one output (f). Each layer contains one of two types of nodes, namely adaptive and fixed nodes. Adaptive nodes, represented by squares, have adjustable parameters, and fixed nodes, denoted by circles, contain fixed parameters. To present the ANFIS architecture, two fuzzy IF-THEN rules based on the first-order Sugeno fuzzy inference system are considered:

Rule 1: **IF** x is A_1 AND y is B_1, **THEN** $f_1 = p_1x + q_1y + r_1$,

Rule 2: **IF** x is A_2 AND y is B_2, **THEN** $f_2 = p_2x + q_2y + r_2$,

where A_i and B_i are fuzzy sets ($i = 1, 2$ is the index of fuzzy rules); f_i are outputs of fuzzy rules; p_i, q_i, and r_i are adaptive parameters determined during the training process. Each layer in the ANFIS architecture is illustrated as below.

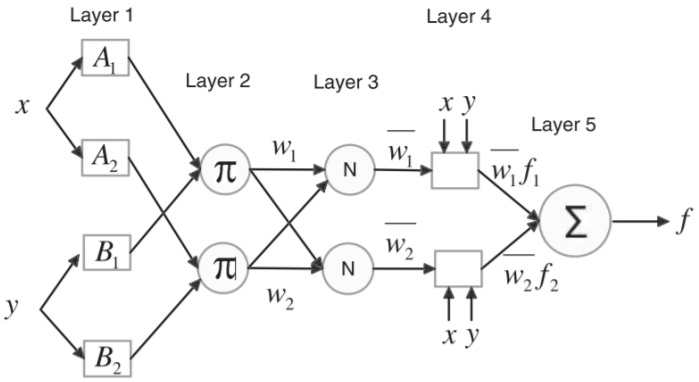


FIGURE 5.3: The ANFIS architecture. Retrieved from [107].

The first layer is the fuzzification layer, where all nodes are adaptive nodes. Outputs θ_j^1 of this layer are the fuzzy membership grades of inputs, which can be represented as:

$$\begin{aligned} \theta_j^1 &= \mu_{A_j}(x), \quad j = 1, 2, \\ \theta_j^1 &= \mu_{B_{j-2}}(y), \quad j = 3, 4, \end{aligned} \tag{5.1}$$

where j is the index of nodes in this layer; μ_{A_j} and $\mu_{B_{j-2}}$ are membership functions associated with fuzzy sets A_j and B_{j-2}, respectively. For the Gaussian membership function, μ_{A_j} and $\mu_{B_{j-2}}$ can be presented as:

$$\begin{aligned} \mu_{A_j}(x) &= e^{-\left((x-b_j)^2/2a_j^2\right)}, \quad j = 1, 2, \\ \mu_{B_{j-2}}(y) &= e^{-\left((y-b_j)^2/2a_j^2\right)}, \quad j = 3, 4 \end{aligned} \tag{5.2}$$

where a_j and b_j are adjustable parameters of the membership functions. Parameters in this layer are referred to as premise parameters.

The second layer uses the AND operator (presented by a circle node labeled π) to fuzzify the incoming signals. All nodes in this layer are fixed, and each node computes the strength of rules (i.e., firing strength). The output of this layer w_j is the product of

the incoming signals:

$$\theta_j^2 = w_j = \mu_{A_j}(x)\,\mu_{B_j}(y), \quad j = 1, 2. \tag{5.3}$$

The third layer is the normalization layer, where all nodes are fixed. The output of this layer $\bar{w}_j$ is the normalization of incoming signals:

$$\theta_j^3 = \bar{w}_j = \frac{w_j}{\sum_{k=1}^{2} w_k}, \quad j = 1, 2. \tag{5.4}$$

In the fourth layer, nodes are adaptive. The output of this layer is the the product of the normalized firing strength $\bar{w}_j$ and first order polynomials of inputs:

$$\theta_j^4 = \bar{w}_j f_j = \bar{w}_j\,(p_j x + q_j y + r_j), \quad j = 1, 2. \tag{5.5}$$

The last layer contains a single fixed node, which sums all incoming signals. The output of this node is represented by

$$\theta_j^5 = y = \sum_{k=1}^{2} \bar{w}_k f_k, \quad j = 1, 2. \tag{5.6}$$

The learning algorithm for ANFIS is the hybrid-learning algorithm, which is the combination of gradient descent and least squares methods. Each epoch of the hybrid learning algorithm is consisted of a forward pass and a backward pass. In the forward pass, premise parameters (i.e., b_j and a_j in the second layer) are kept constant and consequent parameters (i.e., p_j, q_j, and r_j in the fourth layer) are determined by the least squares method. In the backward pass, consequent parameters obtained from the

previous step are kept constant and premise parameters are updated by the gradient descent method. The hybrid learning algorithm ensures a quicker convergence and avoids local minima issue comparing to the gradient descent method. The detailed description of the hybrid-learning algorithm can be found in [109].

5.3.2 Adapt the ANFIS Architecture for OSL Problems

In this work, the ANFIS model is selected to generate the searching strategy that controls the robot searching for the odor source. Steps to adapt the ANFIS model include: define input and output variables; define fuzzy sets for input variables; define fuzzy rules; create and train the ANFIS model.

5.3.2.1 Select Input and Output Variables

In an OSL problem, tens of variables are related with the robot searching behavior. For instance, in the moth-inspired method, sensed odor concentration and wind direction determine whether the robot is in the 'surge' or 'casting' searching phase, and in the Bayesian-inference method, robot positions and wind history records are essential to estimate odor source locations. Since these two methods are employed to generate training data in our method, all mentioned variables are included into inputs of the ANFIS model.

In summary, inputs of the ANFIS model are defined as wind speeds in x and y directions (u_x and u_y), sensed odor concentrations (ρ), robot positions (x and y), and heading measurements (ψ). As mentioned, the robot moving velocity is fixed in this work, thus, the output of the ANFIS model is set as the heading command (ψ_c, see Fig. 5.1(b)). It is noted that wind speeds in x and y directions are calculated from wind speed (u) and

direction (ϕ) measurements:

$$u_x = u \times cos\phi,$$
$$u_y = u \times sin\phi. \tag{5.7}$$

The reason of using u_x and u_y as inputs instead u and ϕ is that the Bayesian-inference method uses u_x and u_y to compute wind advection distances on x and y directions.

5.3.2.2 Define Fuzzy Sets and Fuzzy Rules

To determine the number of fuzzy sets for each input, the complexity of the ANFIS model (i.e., the number of fuzzy rules) needs to be considered to prevent the overfitting (i.e., the complexity of the ANFIS model is greater than the size of training data) problem. Denote the number of inputs to the ANFIS model as I and the number of fuzzy sets of a single input is $M_i, i \in [1, I]$. Then, the total number of fuzzy rules N can be calculated as:

$$N = \prod_{i=1}^{I} M_i. \tag{5.8}$$

In this work, the number of inputs is determined as 6. If three fuzzy sets were defined for a single input, the total number of fuzzy rules would be 729, and if two fuzzy sets were defined for a single input, the number of fuzzy rules drops significantly to 64. It should be mentioned that increasing the number of membership functions per input does not necessarily improve the ANFIS model performance, but usually leads to model overfitting [110]. Thus, two fuzzy sets are defined for each input in the proposed ANFIS model, and the Gaussian function is selected as fuzzy membership functions of all fuzzy sets. Table 5.1 presents labels of fuzzy sets for each input.

For the first-order Sugeno fuzzy inference system, the consequence of a fuzzy rule f_1 can be expressed as:

TABLE 5.1: Fuzzy Sets of Inputs to the ANFIS Model

Inputs	u_x	u_y	ρ	x	y	ψ
Fuzzy Sets	Slow, Fast	Slow, Fast	Low, High	Small, Large	Small, Large	Little, Big

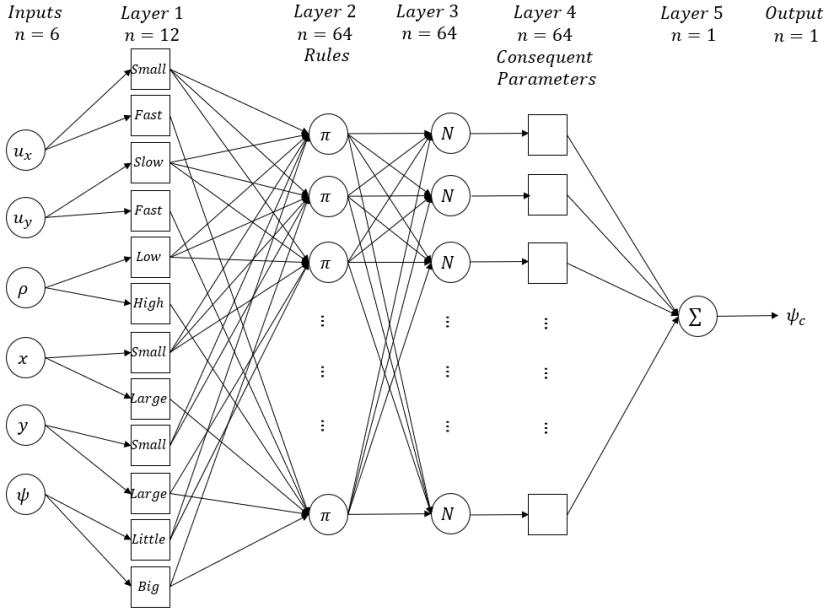


FIGURE 5.4: The adapted ANFIS architecture for OSL problems. The number of nodes in a layer is indicated by n as shown at the top of the diagram.

Rule 1: IF u_x is Slow AND u_y is Slow AND ρ is Low AND x is Small AND y is Small AND ψ is Little, THEN $f_1 = \alpha_{1,1}u_x + \alpha_{2,1}u_y + \alpha_{3,1}\rho + \alpha_{4,1}x + \alpha_{5,1}y + \alpha_{6,1}\psi + \alpha_{7,1}$,

where $\alpha_{i,j}, i \in [1, I], j \in [1, N]$ are the linear parameters in the consequent part of the Sugeno fuzzy inference system.

5.3.2.3 Create and Train the ANFIS Model

In this study, the MATLAB Fuzzy Logic Toolbox is used as the development tool to construct and train the ANFIS model. With the Fuzzy logic toolbox, an ANFIS model can be created and trained with few lines of codes, while training options such as epoch numbers, membership function types and numbers, training algorithms are adjustable.

After the training data set is loaded into the MATLAB *workspace*, the initial ANFIS model is created with the $genfis$ MATLAB function. This function finds the lower and upper bounds of input training data and creates membership functions to evenly cover the discourse of universe of each input. Once the initial ANFIS model is created, the training process begins. In MATLAB, the training command is $anfis$, which reads the training input/output data and generates an ANFIS model that maps inputs to the desired output. Specifically, all training data is fed to a neural network (see Fig. 5.3), in which input parameters are adjusted to minimize training errors. Root mean square error (RMSE) is the function that used to compute the training errors, which is defined as:

$$RMSE = \sqrt{\frac{1}{G}\sum_{j=1}^{G}\left(Y_j - \hat{Y}_j\right)^2}, \tag{5.9}$$

where G is the size of the training data; $\hat{Y}$ is the predicted output generated from ANFIS model; Y is the original output from the training data set, i.e., ψ_c. In summary, the proposed ANFIS architecture is presented in Fig 5.4.

5.3.3 Train ANFIS Models with Training Data Sets

Multiple training data sets are acquired by implementing different olfactory-based navigation methods. As mentioned in the previous section, the moth-inspired and Bayesian-inference methods are selected to generate training data sets. According to the type of the implementing algorithm, three training data sets (moth-inspired, Bayesian-inference, and Fused) and corresponding ANFIS models (MO-ANFIS, BA-ANFIS, and FU-ANFIS) were created. Note that, architectures of three ANFIS models are the same, but their parameters after training with the corresponding training set are varying.

Fig. 5.5 shows plots of RMSE after training ANFIS models with different training data sets. It can be seen that all RMSE plots converge to a constant after around 100 epochs. Specifically, the RMSE of training with the moth-inspired training data set is 0.572 after 100 epochs and slightly drops to 0.571 after 500 epochs. Similar RMSE values can be found for the Bayesian-inference training data set: 0.489 after 100 epochs and 0.488 after 500 epochs. As for the fused training data set, the RMSE is 0.523 when training epoch is 100 and vaguely decreases to 0.522 after 500 epochs. In general, for three training data sets, values of RMSE decrease with the increase of epoch numbers, but after 100 epochs, the change of RMSE is minor. After the training process is complete, trained ANFIS models (MO-ANFIS, BA-ANFIS, and FU-ANFIS) obtained after 500 training epochs are selected and deployed in the simulation with environmental settings $\mathbf{U}_0 = (0.4, 0)$ m/s and $\varsigma = 5$ to verify the validity of the proposed ANFIS method in an OSL problem.

5.3.4 Experiments and Results

To verify the validity of the proposed method, various experiments were conducted. Fig. 5.6 presents searching trajectories generated from implementing three ANFIS models in OSL tests. In these tests, the robot starts at the same position $(60, -40)$ m and adopts a 'zigzag' searching strategy to detect the existence of odor plumes. After the robot detects plumes for the first time, the ANFIS model is activated and controls the robot to search the odor source. As shown in Fig. 5.6, all ANFIS models can correctly locate the odor source in this environment, which demonstrates the capability of ANFIS models to learn other olfactory-based navigation methods.

Specifically, it can be seen in Fig. 5.6(a) that the robot searching trajectory of MO-ANFIS is resembling to the moth-inspired olfactory-based navigation method. Boundaries between 'surge' and 'casting' searching phases are distinct, and these two phases

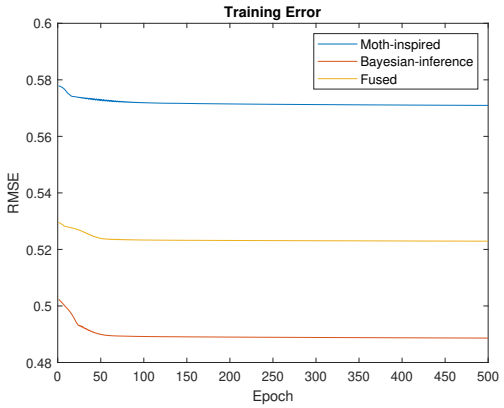


FIGURE 5.5: Plots of RMSE in each epoch with different training sets

are triggered with the plume detection and non-detection events, which is the identical searching logic to the moth-inspired method. The searching trajectory of BA-ANFIS is presented in Fig. 5.6(b). The robot acts like it could estimate the odor source location: after $t = 70$ s, the robot moves in parallel to the plume trajectory, and when it is close to the odor source, the robot correctly turns to the source at $t = 98$. The searching mechanism presented by BA-ANFIS is reminiscent to the Bayesian-inference olfactory-based navigation method, which estimates and surges toward the possible odor source location. For the ANFIS model trained by fused data set, i.e., FU-ANFIS, the searching trajectory is presented in Fig. 5.6(c). In this diagram, the presented searching trajectory is also similar to the moth-inspired method, but the robot moves in a less oscillating way comparing to the MO-ANFIS trajectory. Comparing to the BA-ANFIS, the robot with FU-ANFIS moves less aggressively in the upwind direction and takes a slightly longer time (i.e., 100 s) to find the odor source.

Comparing searching results of three ANFIS models, the performance of the FU-ANFIS model is better than the others. Although the searching time of FU-ANFIS is marginally longer than BA-ANFIS, the searching trajectory of the FU-ANFIS model is more effective. It is because source estimations based on Bayesian-inference method usually are not reliable before the robot gains enough information about the odor source. It can

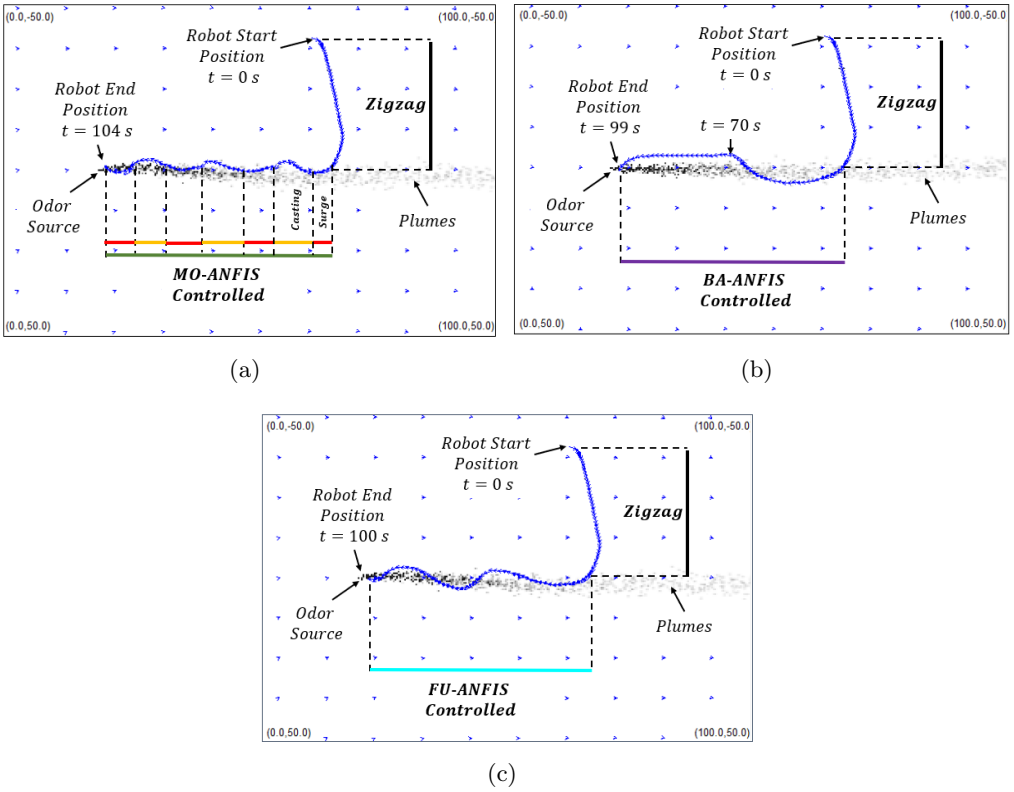


FIGURE 5.6: Search trajectories of implementing different ANFIS models in the simulation with environmental settings $\mathbf{U}_0 = (0.4, 0)$ m/s and $\varsigma = 5$. Search phases are indicated by different colors, where black represents the 'zigzag' search phase; green, purple, and cyan indicate MO-ANFIS, BA-ANFIS, and FU-ANFIS controlled phases, respectively; red and yellow represent 'surge' and 'casting' searching phases, respectively. (a) MO-ANFIS. (b) BA-ANFIS. (c) FU-ANFIS.

be seen that the robot can always detect plumes by following the FU-ANFIS trajectory, which is an essential behavior for the robot to acquire odor source information. The robustness of the FU-ANFIS model in OSL tests is further investigated in the next section.

5.3.5 Test ANFIS Models with Various Searching Conditions

Experiment results of implementing the FU-ANFIS with different searching conditions, including varying initial searching positions and environmental settings, are presented in this section.

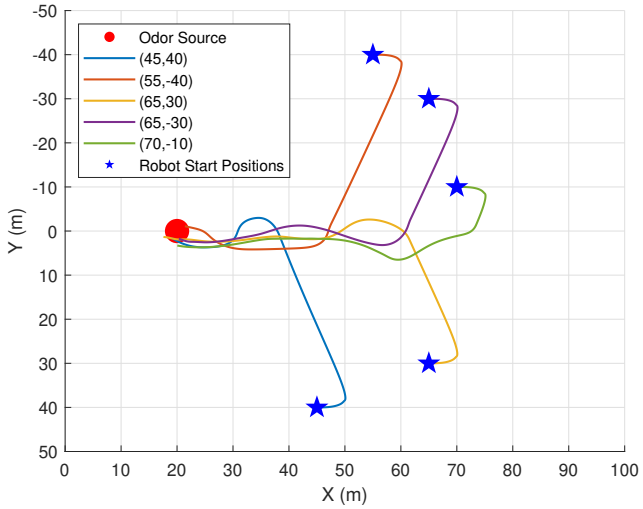


FIGURE 5.7: Searching trajectories generated from different initial robot positions with the FU-ANFIS model. The initial start position of each trail is indicated in the diagram legend.

5.3.5.1 Varying Robot Starting Positions

In this group of tests, the robot starts the OSL task at different initial positions with the same environmental settings $\mathbf{U}_0 = (0.4, 0)$ m/s and $\varsigma = 5$. Fig. 5.7 shows the searching trajectories of implementing the FU-ANFIS model with varying initial robot positions. As shown in the diagram, all trails terminate at the actual odor source location, i.e., the robot correctly locate the odor source location, which indicates the validity of the proposed ANFIS model with varying initial start positions.

5.3.5.2 Varying Environmental Settings

To evaluate the performance of the FU-ANFIS model, the moth-inspired and Bayesian-inference olfactory-based navigation methods are also implemented and compared. Each navigation method is implemented in five different environments, and the corresponding searching time of each method is presented in Table 5.2.

TABLE 5.2: Environmental Settings and Searching Time of Different Navigation Methods

Test	Mean Wind Velocity $\mathbf{U}_0$ (m/s)	Gaussian Noise Variance ς	FU-ANFIS Model-Based Method (s)	Moth-Inspired Method (s)	Bayesian-Inference Method (s)
Test 1	[0.1, 0]	3	92	93	132
Test 2	[1,0]	3	96	95	121
Test 3	[0.8, 0.1]	3	99	93	139
Test 4	[1.5, 0.1]	3	95	141	153
Test 5	[2, 0.3]	3	Failed	Failed	137

Performances of FU-ANFIS and moth-inspired methods are similar in terms of the searching time in Test 1, 2, and 3, where the robot is in laminar flow environments. In a more turbulent environment, e.g., Test 4, the moth-inspired method requires a longer searching time (141 s) than the FU-ANFIS method (95 s). Both FU-ANFIS and moth-inspired methods failed to locate the odor source in a highly turbulent environment (Test 5). The averaged searching time of FU-ANFIS method for Test 1-4 is 95.5 s. Comparing to moth-inspired method, which achieves the averaged searching time as 105.5 s for same tests, the FU-ANFIS method is more efficient. Comparing to the Bayesian-inference method, it can be concluded that FU-ANFIS method is more efficient in laminar and slightly turbulent environments (Test 1-4), but the Bayesian-inference method is more effective in highly turbulent environments.

5.3.6 Discussions

Although experiment results of implementing the ANFIS model in OSL tests are satisfied, we should also see the limitation of this method: the performance of the trained ANFIS model is highly depending on the quality and quantity of the training data set. To obtain a versatile ANFIS model that can succeed to locate odor sources in more extreme environments (e.g., Test 5) needs more training data sets. A preferred

training data set should cover more searching situations, such as searching in various environments (both laminar and turbulent flow fields) and different initial positions. In addition, besides moth-inspired and Bayesian-inference olfactory-based methods, more odor searching strategies could be considered to generate training data sets.

5.4 Train ML Models with DNNs

5.4.1 Problem Formulation

The main objective of this method is to obtain a DNN model that guides a plume tracing robot to locate an odor source in an unknown environment. To achieve this goal, the DNN model is trained to calculate suitable robot commands $\mathbf{C}$ based on robot states $\mathbf{S}$:

$$\mathbf{C} = \mathcal{F}_\theta(\mathbf{S}). \tag{5.10}$$

This DNN model is parametrized by a parameter vector θ, and the optimal θ is found during the process of supervised training, which minimizes the difference between outputs of the DNN and the ones demonstrated by expert methods.

5.4.2 Defining Inputs and Outputs of DNNs

As mentioned, two expert methods, namely moth-inspired and Bayesian-inference methods, are employed to generate training data sets. To learn expert methods, DNNs should be offered with similar input information. In the moth-inspired method, odor concentrations (ρ) and wind directions (ϕ) determine whether the robot is in the 'surge' or 'casting' search phase, and for the Bayesian-inference method, robot positions (x and y), wind speeds (u_x and u_y), and algorithm running time (t) are essential to estimate

TABLE 5.3: Definitions of Variables

Symbols of Variables	Definitions of Variables
t (s)	Algorithm running time
u (m/s)	Wind speed at the robot position
ϕ (rad)	Wind direction at the robot position
ρ (mmpv)	Odor concentration at the robot position
x (m)	Robot horizontal position
y (m)	Robot vertical position
ψ (rad)	Robot heading angle
v (m/s)	Robot speed
v_c (m/s)	Robot speed command
ψ_c (rad)	Robot heading command

mmpv: million molecules per cm^3

odor source locations. All aforementioned variables should be included in DNN's input state, therefore, the input state vector $\mathbf{S}$ is defined as:

$$\mathbf{S} = (t, u_x, u_y, \rho, x, y, v_x, v_y) \tag{5.11}$$

where u_x, u_y, v_x and v_y are wind and robot speeds in x and y directions, respectively. To control a mobile robot on a 2-D plane, only speed and yaw angle commands (v_c and ψ_c) are needed. Thus, DNN's outputs are defined as:

$$\mathbf{C} = (v_{c,x}, v_{c,y}) \tag{5.12}$$

where $v_{c,x}$ and $v_{c,y}$ are robot velocity commands on x and y directions, respectively.

It should be mentioned that we convert angle-related variables, including wind directions (ϕ), robot yaw angles (ψ), and yaw angle commands (ψ_c), to vector forms in $\mathbf{S}$ and $\mathbf{C}$:

$$\begin{cases} u_x = u\cos\phi, v_x = v\cos\psi, v_{c,x} = v_c\cos\psi_c \\ u_y = u\sin\phi, v_y = v\sin\psi, v_{c,y} = v_c\sin\psi_c \end{cases}. \tag{5.13}$$

This is because angles do not make a good DL model input: one angle could refer to

two different values such as $-\pi$ and π, and angles should not matter if the corresponding speed is zero. For the easy reference, Table 5.3 lists variables and corresponding definitions in $\mathbf{S}$ and $\mathbf{C}$.

5.4.3 Training Data Specifications

To collect training data, around 6000 OSL trials are conducted for each expert method. In an OSL trial, a data tuple $\gamma_t = (\mathbf{S}_t, \mathbf{C}_{exp,t})$ that consists of input states $\mathbf{S}_t$, which are obtained from robot sensor measurements, and expert commands $\mathbf{C}_{exp,t}$, which are produced by the implemented expert method, is recorded at every time t during the plume tracing process. An OSL trial is considered as complete if the robot reaches the odor source location or the algorithm running time is beyond the time limit, i.e., 400 s in this work. It should be mentioned that we consider an odor source has been found if the robot is in vicinity of it. In real-world applications, this step, i.e., source declaration, could be complete with aids of external sensors such as cameras, which could recognize an odor source from a close distance.

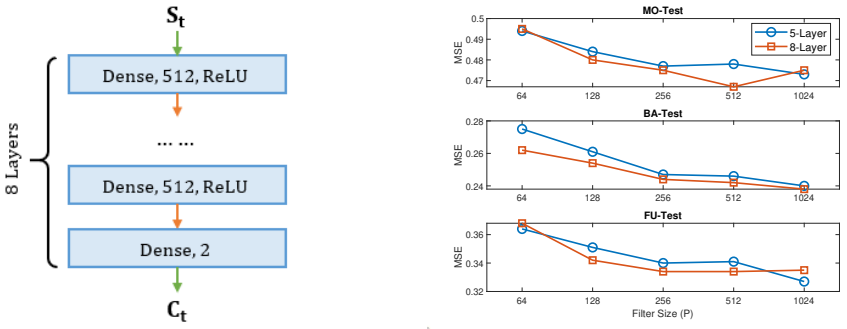
Depending on the type of the expert method, two training data sets, namely MO-Train (obtained from the moth-inspired method) and BA-Train (obtained from the Bayesian-inference method), are acquired. Previous experiment data [111] reveals that bio-inspired methods are more efficient (i.e., require less searching time) in laminar flow environments while engineering-based methods outperforms the counterpart in turbulent flow environments in terms of the search success rate. Thus, to learn benefits from two expert methods, two training data sets are combined to form a fused data set, which is named as FU-Train.

Only 80% of training data is used to train DNN models, while 10% of the remaining data, termed testing data set, is used to test DNN models after the training, and the last 10% data, termed validation data set, is used to compute validation errors during the training process: the training process is terminated once the validation error is not improved in 20 episodes. The validation error is defined as mean absolute errors (MAEs) in this work, i.e., $1/n \cdot \sum_{i=1}^{n} |\mathcal{F}_\theta(\mathbf{S}_i) - \mathbf{C}_{i,exp}|$ where n is the size of validation data set.

5.4.4 Design DNNs for OSL Problems

Two types of neural networks, i.e., FNN and CNN, are selected for the representation of $\mathcal{F}_\theta$. The motivation for choosing FNN is that we want to use a simple DNN structure to investigate the viability of implementing DL approaches on OSL problems. Besides, the intuitive FNN could also be employed as the baseline to evaluate the performance of other types of DNN models in the OSL problem. Fig. 5.8(a) presents the structure of a FNN model. To determine the optimal numbers of hidden layers and filter sizes, varying values are investigated. Fig. 5.8(b) shows the mean square errors (MSEs) of implementing different structure FNNs with varying hidden layers and filter sizes on testing data sets. It can be observed that larger models (i.e., more layers and filters on each layer) achieve better performances (i.e., lower MSEs) but overfit when the model is too complicated (i.e., the MSE increases). Based on plots, the FNN with 8-layer and 512 filters is selected for implementing in OSL tests.

Due to the characteristics of FNN structure, the sensor data history (i.e., $\mathbf{S}_{t-1}$, $\mathbf{S}_{t-2}$, ...) is ignored in calculating FNN outputs. In terms of learning expert methods, the FNN may learn the moth-inspired method well since this expert method does not consider history information as well. However, for the other expert method, i.e., Bayesian-inference method, the sensor data history is the necessary information to estimate the odor source



(a) (b)

FIGURE 5.8: (a) The structure of the proposed FNN. Labels inside a blue layer represent the layer type, filter size, and activation function, respectively. A dense layer indicates a fully-connected neural network; (b) MSEs of FNNs with varying filter sizes and hidden layers on testing data sets

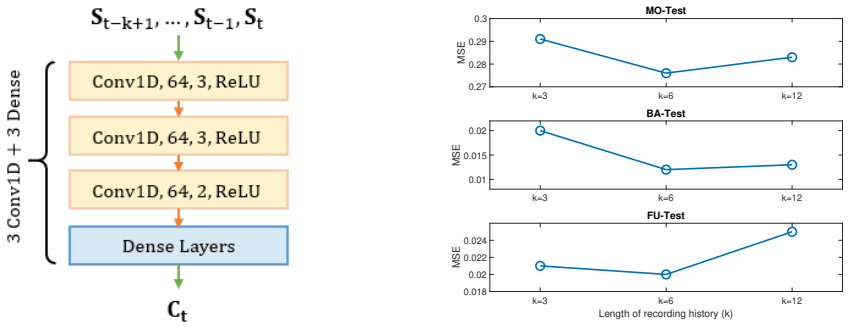
location. To address this issue, a CNN model is employed to process sensor data and produce robot commands.

As presented in Fig. 5.9(a), the proposed CNN not only considers sensor data at the current time but also from previous k time steps. Specifically, three convolution layers are employed to extract features from sensor data history. Then, the extracted features are fed to three dense layers to produce robot commands. To find the optimal value of k, different numbers are investigated. Fig. 5.9(b) shows MSEs of CNNs with varying k on testing data sets. It can be observed that MSE is lowest on all testing data sets when $k = 6$. Therefore, we choose this CNN structure to implement in later OSL tests.

The mathematical representation of the CNN model can be presented by:

$$\mathbf{C}_t = \mathcal{F}_{\theta_{CNN}}(\mathbf{S}_{(t-k)\sim t}, \mathbf{C}_{(t-k)\sim(t-1)}), \tag{5.14}$$

where θ_{CNN} represents the parameter vector of the CNN model, k is the length of recording window, $\mathbf{S}_{(t-k)\sim t} = (\mathbf{S}_{t-k}, \mathbf{S}_{t-k+1}, ..., \mathbf{S}_t)$ is a vector of sensor data from time $t-k$ to t, and $\mathbf{C}_{(t-k)\sim(t-1)} = (\mathbf{C}_{t-k}, \mathbf{C}_{t-k+1}, ..., \mathbf{C}_{t-1})$ is a vector of historical robot



(a) (b)

FIGURE 5.9: (a) The structure of the proposed CNN. Labels inside a yellow layer represent layer type, filter size, kernel size, and activation function, respectively. The Conv1D layer means a convolutional neural network. The last 'Dense Layers' block contain 3 consecutive dense layers with 512, 512, and 2 filters, respectively; (b) MSEs of CNNs with the varying length of recording history on testing data sets

commands from time $t-k$ to $t-1$.

5.4.5 Training DNN Models

In this work, we employ the supervised learning [112] as the training algorithm, which aims to find the optimal parameter vector θ^* that minimizes the loss function J. The loss function J is defined as the mean square error between DNN outputs $\mathcal{F}_\theta(\mathbf{S})$ and expert demonstrations $\mathbf{C}_{exp}$, which can be represented as:

$$J\left(\Gamma_B\right) = \frac{1}{N_B} \cdot \sum_{i=j}^{j+N_B} \left(\mathcal{F}_\theta\left(\mathbf{S}_i\right) - \mathbf{C}_{i,exp}\right)^2, \tag{5.15}$$

where Γ_B is a mini-batch that contains N_B (32 in our work) samples from a training data set. The gradient of the cost function with respect to model parameters is calculated using the backpropagation algorithm [113], and the optimization algorithm that updates model parameters is the Adam optimizer [114].

To train FNNs, the order of training samples is randomized to reduce the temporal correlation in training data sets. This procedure is skipped in the process of training

CNNs since the proposed CNN produces robot commands based on time series data. The training process is considered as complete if one of the following two conditions is satisfied: 1) the training epoch reaches the limit (i.e., 500 in implementations); 2) the validation error does not improve in 20 consecutive epochs. Google® TensorFlow [115] is employed as the framework to construct and train DNN models. Training the proposed FNN and CNN with the FU-Train data set (1.6 million data tuples) on an Intel® i7-8750 CPU with the Nvidia® GeForce GTX 1070 GPU acceleration takes around 4 hours and 2 hours, respectively.

5.4.6 Sample OSL Trials

To exam the effectiveness of DNN models after training, the author first implements the proposed FNN and CNN, trained with the fused training data set (i.e., FU-Train), in sample OSL trials, where the mean flow velocity of the environment is $\mathbf{U}_0 = (1, 0)$ m/s and the variance of Gaussian white noises is $\varsigma = 3$. The robot starts at $(60, -40)$ m, and the odor source is located at $(20, 0)$ m.

Fig. 5.10 shows search trajectories of two expert methods (i.e., moth-inspired 5.10(a) and Bayesian-inference methods 5.10(b)) and the proposed DNN models (i.e., FNN and CNN) in the sample OSL trial. In these plume tracing methods, the robot first adopts a 'zigzag' search strategy [33] to find the existence of plumes, and after the first plume detection, the corresponding navigation method is activated, which guides the robot to trace plumes. In this laminar flow environment, the moth-inspired method achieves a shorter search time compared to the Bayesian-inference method (93 s vs 126 s). This is because in a laminar flow environment, odor plumes form a stable and continuous trajectory, in which the 'surge' behavior of the moth-inspired method is more effective

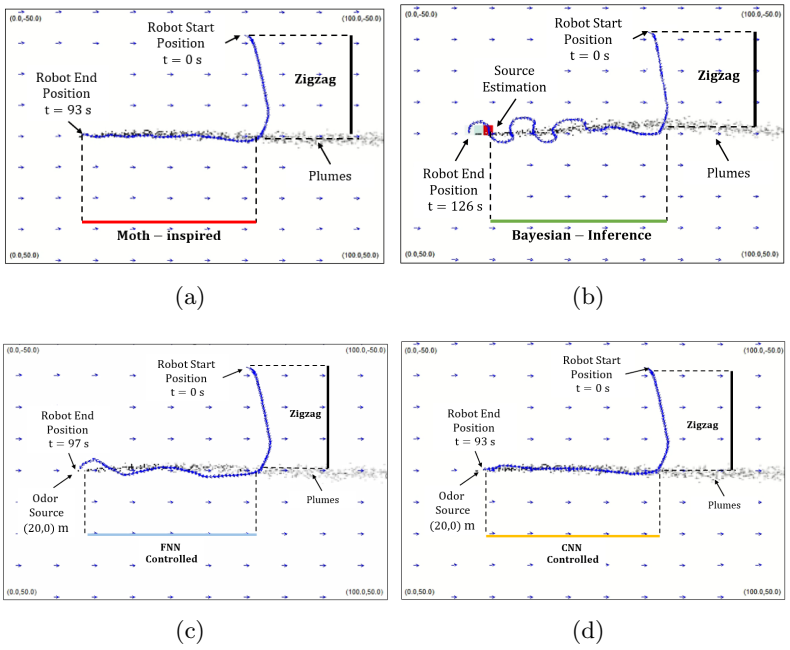
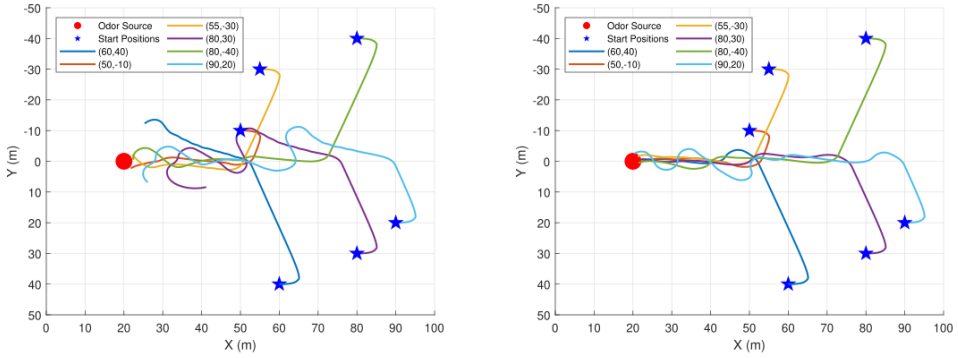


FIGURE 5.10: Search trajectories of expert methods and DNNs in the sample OSL trial, where (a) Moth-inspired method, (b) Bayesian-inference method, (c) FNN, and (d) CNN. The robot trajectory is presented by the blue curve, where the robot initial and end positions are indicated in diagrams. The duration of each navigation method is labeled by different color bars, where black represents zigzag; red represents moth-inspired; green represents Bayesian-inference; light blue represents FNN; yellow represents CNN.

to quickly trace up-wind and locate the odor source compared to engineering-based methods.

It can be observed in Fig. 5.10(c) and Fig. 5.10(d) that both FNN and CNN can correctly locate the odor source in the sample OSL trial. In addition, both trajectories is similar to the one produced by the moth-inspired method, which is a preferred navigation method in this type of environment (i.e., laminar flow environments). Specifically, the FNN search trajectory shows a 'casting' alike behavior to traverse plumes when plumes are absent, while the proposed CNN acts analogously to the 'surge' behavior that controls the robot consistently detecting plumes by moving up-wind. Comparing FNN and CNN search trajectories, the CNN generates a smoother trajectory and finds the odor source within a shorter search time (97 s vs 93 s). Besides, it can be seen that the robot



(a) (b)

FIGURE 5.11: Search trajectories generated by (a) FNN and (b) CNN in OSL tests with varying robot initial positions

can consistently detect plumes by following the trajectory produced by CNN, which is beneficial for the plume-tracing robot to acquire adequate odor source information.

5.4.7 Varying Search Conditions

To investigate the generalization of the proposed DNN models, they are implemented in OSL tests with different search conditions, including varying robot initial positions, odor source locations, and environmental settings.

5.4.7.1 Varying Robot and Source Positions

In this group of tests, the proposed DNN models are evaluated in environments with different robot initial positions and odor source locations. Fig. 5.11 presents the search trajectories of the proposed FNN and CNN with six different robot initial positions that are unseen for DNN models in the training process. It can be seen that without considering the sensor data history, the proposed FNN can barely adapt to new search conditions, where only three out of six trajectories correctly find the odor source. On the other hand, given the same training data sets, the CNN is more effective than FNN

TABLE 5.4: Environmental Settings and Search Time of Different Navigation Methods

Environment Index	Mean Wind Velocity $\mathbf{U}_0$ (m/s)	Gaussian Noise Variance ς	Test Index	Moth-inspired Method (s)	Bayesian-inference Method (s)	The Proposed FNN (s)	The Proposed CNN (s)
Env. 1	$(1, 0)$	3	Test 1	93	123	97	93
			Test 2	91	116	98	103
			Test 3	96	139	123	102
Env. 2	$(1, 0)$	10	Test 1	93	151	-	129
			Test 2	128	125	-	95
			Test 3	94	128	-	110
Env. 3	$(1, -0.4)$	8	Test 1	-	129	-	138
			Test 2	-	116	-	140
			Test 3	-	139	-	-
Env. 4	$(2.5, 0)$	10	Test 1	90	110	-	106
			Test 2	94	176	-	104
			Test 3	95	129	-	145
Env. 5	$(2.5, 0.4)$	12	Test 1	239	95	-	159
			Test 2	-	100	-	160
			Test 3	-	133	-	108

-: Fail to locate the odor source within 400 s.

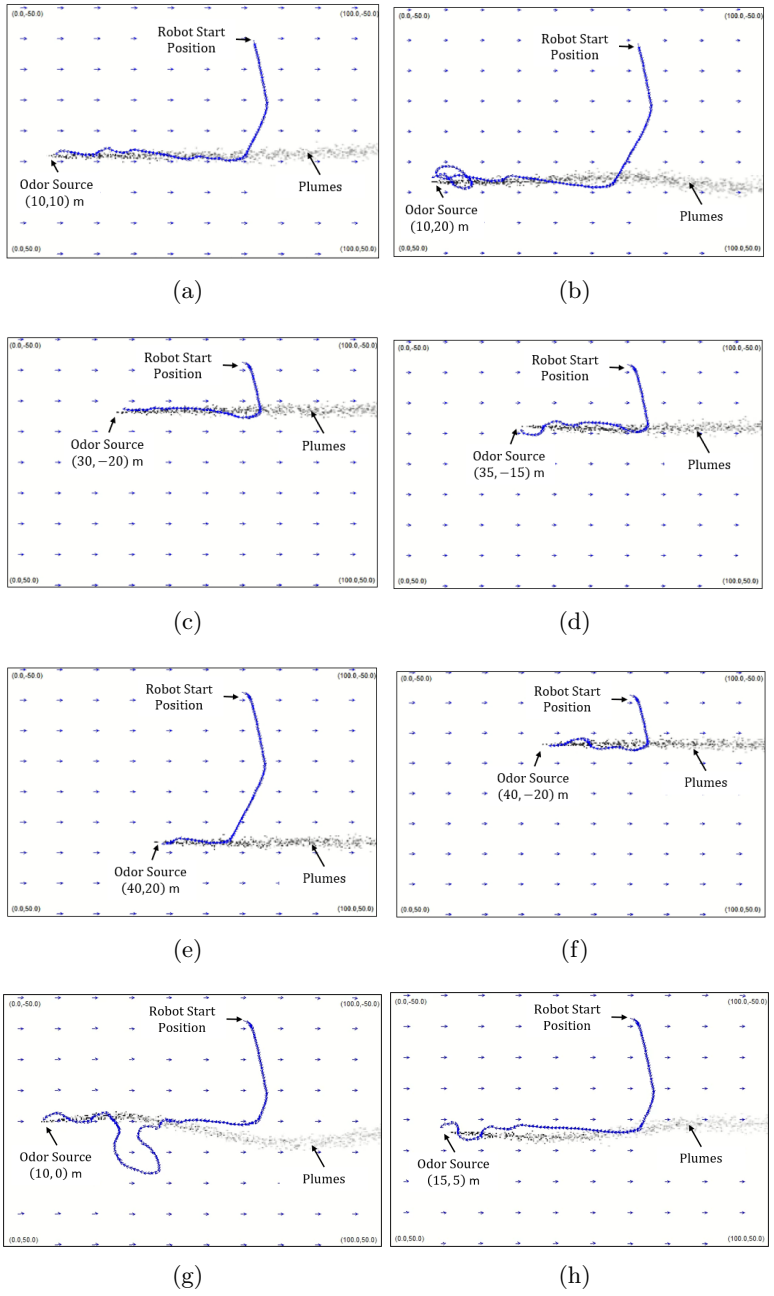


FIGURE 5.12: Search trajectories of the proposed CNN in environments with varying odor source locations, where (a) (10,10) m, (b) (10,20) m, (c) (30,-20) m, (d) (35,-15) m, (e) (40,20) m, (f) (40,-20) m, (g) (10,0) m, and (h) (15,5) m.

in this group of tests, where all trajectories terminate at the odor source location, i.e., the robot correctly finds the odor source in six tests under the direction of the proposed CNN model.

Then, the DNN models are implemented in environments with different the odor source

locations. It was discovered in previous test results that if the odor source location is fixed in the training data sets, a DNN model can memorize the odor source location and skip the plume tracing process, i.e., the robot controlled by the DNNs will proceed directly to the fixed odor source location instead of tracing plumes. To address this problem, the odor source location is varying in each OSL trial in the training data sets.

To verify the generalization of the proposed DNN models on varying odor source locations, eight different odor source locations that are unseen to DNN models, i.e., the ones that are not included in training data sets, are tested. Fig. 5.12 presents search trajectories of implementing the proposed CNN in these tests. We can observe that the robot successfully finds the odor source in these eight trials, which demonstrates the validity of the proposed CNN model for finding varying odor source locations. The proposed FNN fails to find the odor source in this group of tests. One possible approach to improve the FNN's search performance is to enlarge the training data set, i.e., make training data set cover more search examples.

5.4.7.2 Varying Environmental Settings

In this group of tests, environmental settings, i.e., mean flow speed $\mathbf{U}_0$ and Gaussian noise variance ς, are varying to produce different airflow environments. To evaluate the search performance of the proposed DNNs, two expert methods are also implemented and compared in this group of tests. Table 5.4 presents five different environments and the corresponding search time of four navigation methods. Each navigation method is repeatedly performed three times in every environment. It should be mentioned that environmental settings in Env. 2-5 are unseen for DNN models during the training process.

TABLE 5.5: Statistical Results of Repeated Tests and the Comparison of Different Navigation Methods

	Total Tests	Successful Tests	Success Rate	Averaged Search Time (s)
Moth-inspired Method	15	10	67%	111.3
Bayesian-inference Method	15	15	100%	127.3
The Proposed FNN	15	3	20%	106.0
The Proposed CNN	15	14	93%	112.8

As presented in Table 5.4, the proposed FNN can only find the odor source in a laminar flow environment, i.e., Env. 1, while the proposed CNN succeeds in almost all tests except Test 3 in Env. 3. This result confirms that without considering the sensor data history as inputs, the FNN structure can hardly learn an effective plume tracing method, which is not suitable for an OSL problem. By contrast, search results of CNN in this group of tests indicate that the proposed CNN can learn an effective navigation strategy and apply the learned knowledge on new environments.

Table 5.5 presents the statistical results of different navigation methods on varying environment tests. Compared to expert methods, the proposed FNN barely succeeds in these tests, As for the proposed CNN model, it outperforms the moth-inspired method in terms of the success rate (93% vs 67%) and achieves a shorter averaged search time than the Bayesian-inference method (112.8 s vs 127.3 s). Additionally, although the success rate of the proposed CNN is slightly lower than the Bayesian-inference method (93% vs 100%), the CNN method is preferred to be implemented on real-world applications due to the predictable query time.

As mentioned, the computational time of the Bayesian-inference method grows significantly with the increase of the size of the search area [51]. When the search area becomes complex and large, the Bayesian-inference method is not suitable for implementing on

robotic platforms due to the long querying time. On the other hand, the computational cost of the proposed CNN is fixed, which is independent with the size of the search area. Thus, the proposed CNN is preferable for real-world applications.

5.4.8 Discussions

From experiment results, we observe that the DNN structure is a critical factor that affects the DNN search performance in OSL tests. Given the same training data, the FNN structure is not as effective as CNN in experiments (e.g., Section 5.4.7). Because plume-tracing is a continuous process, the search context is important for the robot to make decisions. The CNN structure, which generates robot commands based on previous sensor data, is suitable for the plum-tracing process. Another essential factor is the quality and quantity of training data sets. To achieve satisfying search results in unseen environments, a DNN model should be trained with data set that covers sufficient search examples in both laminar and turbulent airflow environments. To improve this work, other expert methods could also be considered to generate training data sets, and more types of DNN models can be investigated and implemented in OSL problems.

5.5 Summary of this Chapter

This chapter presents ML and AI-based olfactory-based navigation algorithms for navigating a mobile robot to find an odor source in unknown environments. The ANFIS model and two types of DNN models, namely FNN and CNN, are trained with traditional olfactory-based navigation methods. After the training, the proposed ML models can guide a plume-tracing robot to locate an odor source based on the robot sensor data. Experiment results show that the proposed ANFIS model is feasible in locating odor

source with different searching conditions. For DNN models, simulation results show that given the same training data, CNN performs better than FNN on unseen search environments. Compared to traditional methods, experiment results show that the proposed CNN is more desired for autonomous OSL problems since it achieves a comparable search performance with an engineering-based method but is more stable and requires less computational time. However, it should also be mentioned that more training data that covers versatile searching situations is required to improve the searching efficiency of ML models.

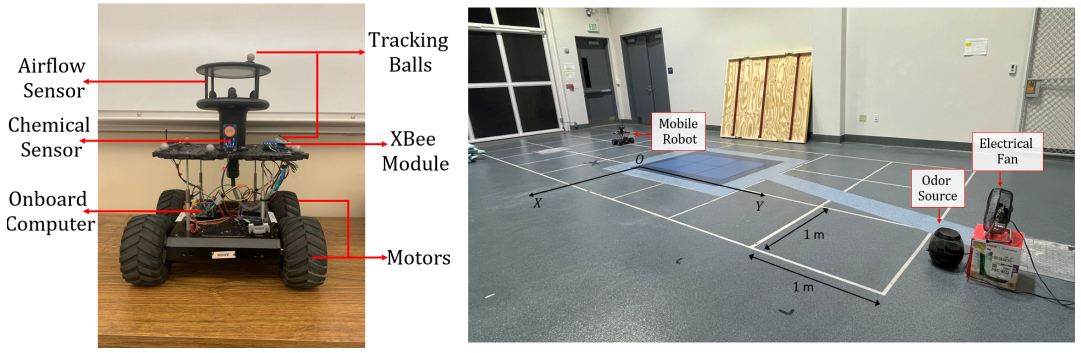
Chapter 6

On-Vehicle Experiments

This chapter presents on-vehicle experiments of the proposed olfactory-based navigation algorithms, including the adaptive behavior based method (i.e., Chapter 3), the reinforcement learning based (RL-based) method (i.e., Chapter 4), and the deep learning based (DL-based) method (i.e., Chapter 5). Experiment setup, design, and results will be introduced, and the search performance of the proposed algorithms will be compared to traditional methods, including the traditional moth-inspired method [37] and the Bayesian-inference method [51].

6.1 Experiment Setup

To verify the effectiveness of the proposed algorithms, the proposed methods were implemented on a mobile robot to find the odor source in real airflow environments. As presented in Fig. 6.1(a), a mobile robot was constructed and employed as the robotic agent, which carries a comprehensive sensor suite for perceiving the environment and wireless communication modules for data transmission. The search area, as presented



(a) (b)

FIGURE 6.1: (a) The mobile robot used in this work. This robot is equipped with an airflow sensor for measuring wind speeds and directions; a chemical sensor for detecting odor plumes; Xbee modules for wireless communication; an onboard computer for processing sensor data. (b) The experiment setup.

in Fig. 6.1(b), contains an odor source and an electrical fan. In an OSL test, the odor source location is hidden to the robot, and the implemented olfactory-based navigation method directs the robot to find the odor source.

6.1.1 Experiment Field

Experiments were conducted in the indoor autonomous robots testing lab at the Embry-Riddle Aeronautical university. The lab is divided into two areas, including a search area where the robot can move and an operation area for accommodating the ground station. As shown in Fig. 6.2(a), the size of the search area is 9×4 m^2, containing an odor source and an electrical fan. It should be mentioned that the size of search area is defined according to the longest sensing distance of the employed chemical sensor. In addition, the lab contains a localization system (i.e., Vicon), which provides accurate positions and orientations of the robot, facilitating the robot control and navigation process. The main airflow direction is created by the electrical fan, but the overall airflow field is turbulent since there are four vents mounted on the wall that occasionally blow winds to maintain the room temperature. The ethanol vapor was employed as the odor source since it is

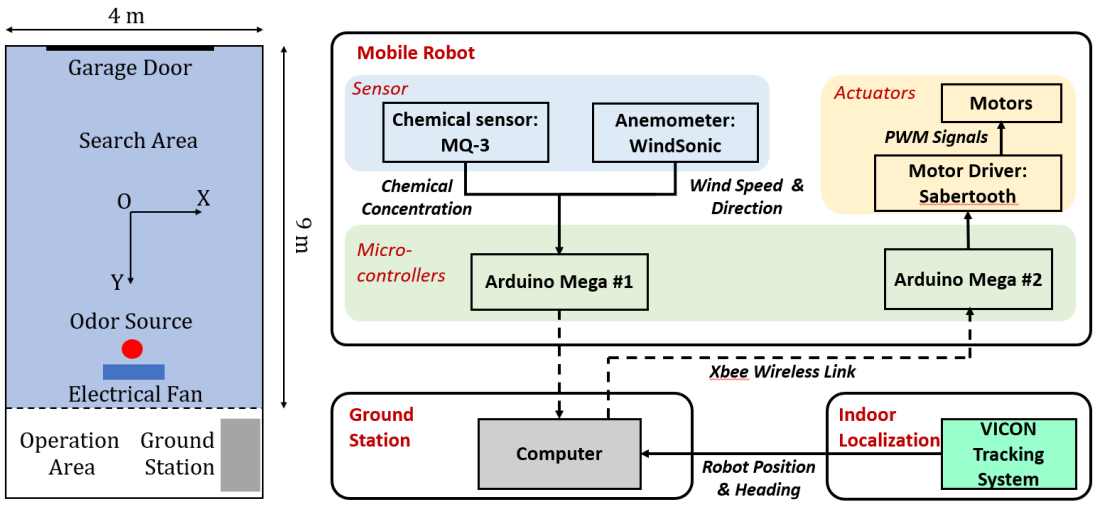


FIGURE 6.2: (a) Search area (b) System configuration. This system contains three main components, including mobile robot, ground station, and indoor localization system. The solid connection line represents physical cables, and the dotted connection line represents wireless link.

minimally toxic and commonly implemented in OSL research [77]. To accelerate the odor dispersion, ethanol was put in a humidifier to consistently release odor plumes.

6.1.2 Mobile Robot

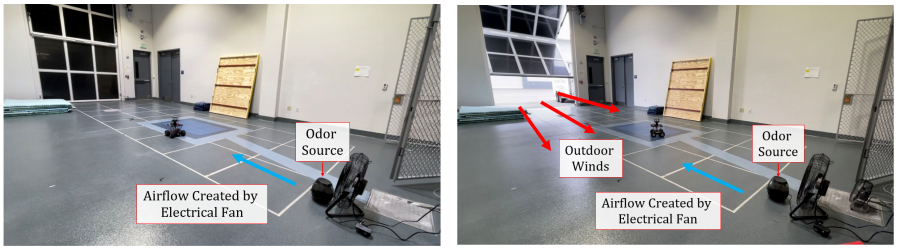
Fig 6.2(b) shows the configuration of the implemented system. There are three main components, including a mobile robot, a ground station, and an indoor localization system. The robot is equipped with a chemical sensor (MQ-3, Waveshare) and an anemometer (WindSonic, Gill Instruments). Both sensors are connected to a micro-controller (Arudino Mega, Arduino) for fetching sensor measurements. The second on-board micro-controller controls robot motors via a motor driver (Sabertooth, Dimension Engineering). Two micro-controllers can communicate with the ground station via a wireless communication network (Xbee, Digi international). The Vicon tracking system (Vicon Inc.) is employed to determine indoor positions, which sends robot positions

and orientations to the ground station via an Ethernet cable. The response time of all sensors were set to 0.25 s.

The robot starts an OSL test in downwind areas. During an OSL test, the robot sends sensor measurements to the ground station. Since the chemical sensor has a long recovery time, i.e., the concentration measurements raise up quickly when the sensor detects odor plume and go down slowly when odor plume is not detected, an adaptive concentration threshold [52] is employed to determine the odor detection and non-detection events. The navigation algorithms are implemented in the ground station to calculate robot's heading commands, which will be transmitted to the mobile robot via the wireless communication network. To save the search time and simplify the control problem, the robot moves in a constant speed, i.e., 0.15 m/s, thus, only heading command is needed to control the robot in a 2-D plane. Then, the robot moves toward the target heading and collects information at the new location. The transmission time between the robot and ground station is negligible due to the short transmission distance. These processes repeat until the robot finds the odor source, i.e., the distance between the odor source and robot is less than 0.5 m.

6.2 Experiment Design

Hundreds of OSL tests were conducted to evaluate the performance of the proposed navigation methods. Among these tests, the mobile robot was placed at different initial positions and tested in different airflow fields. As shown in Fig 6.3(a), in the laminar flow environment, the airflow field is mainly generated by the electrical fan. To create a turbulent airflow field, the garage door at the back of the lab is opened as presented in Fig 6.3(b), allowing outdoor turbulent winds blowing into the search area.



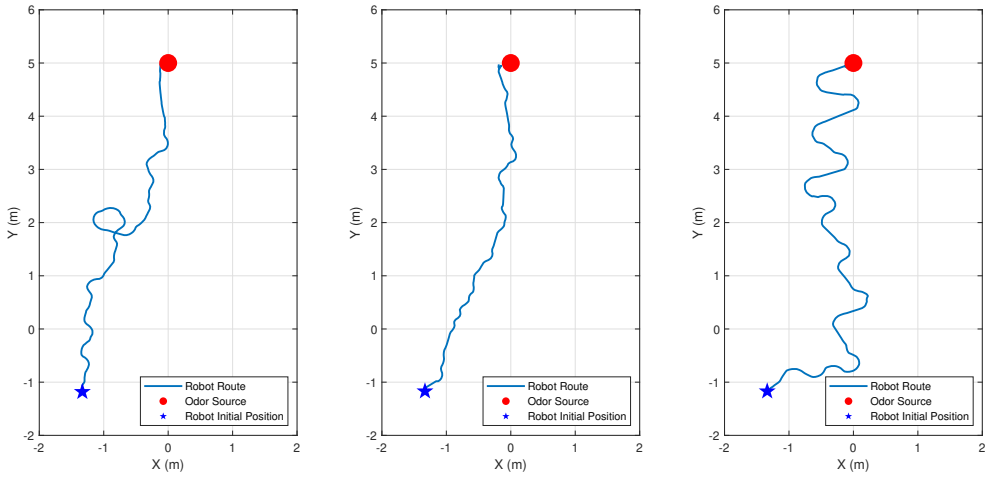
(a) (b)

FIGURE 6.3: Two airflow conditions. (a) Laminar airflow environment. The garage door is closed, creating an enclosed search space. The main airflow direction is created by the electrical fan. (b) Turbulent airflow environment. The garage door is opened, allowing outdoor winds blowing inside the search area.

Besides, the traditional moth-inspired [37] and engineering-based (i.e., the Bayesian-inference method [51]) methods were employed as expert methods to generate training data for the deep learning-based methods. Each traditional method was repeatedly performed around 120 times, where the robot initial position, odor source location, and airflow field varied to diversify the training data set. During the plume tracing process, onboard sensor readings and robot heading commands were recorded to generate training data sets. Depending on the implemented expert method, training data sets can be categorized as MO-Train (i.e., the moth-inspired method) and BA-Train (i.e., the Bayesian-inference method). Additionally, 10% of the collected data will be assigned to testing data sets, i.e., MO-Test and BA-Test.

6.3 Experiment Results

In this section, search results of the proposed navigation methods are presented. To compare the search performance of the bio-inspired and engineering-based methods, search results of adaptive behavior-based method and RL-based method are presented in the same section (i.e., Section 6.3.1), and the search results of DL-based methods



(a) (b) (c)

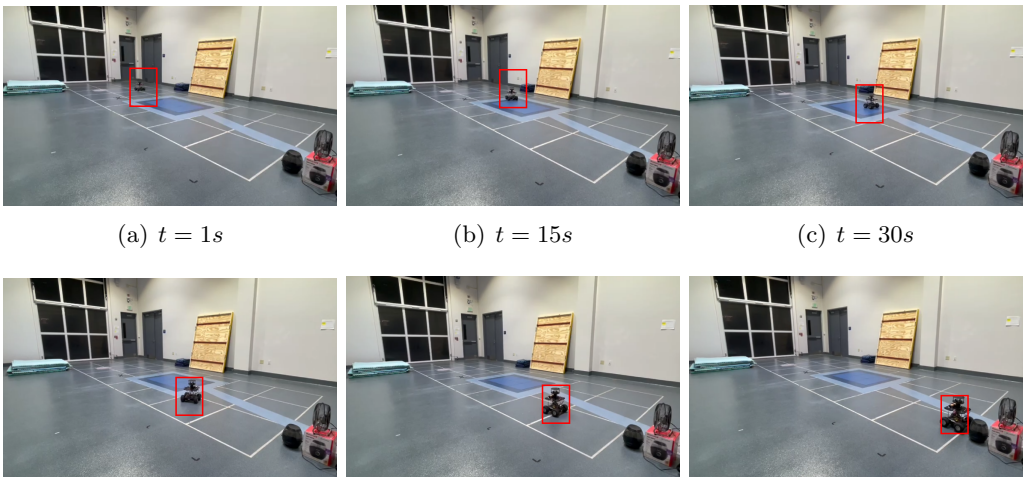
FIGURE 6.4: Sample runs of three algorithms in sample trails. The blue star indicates the robot initial position $(-1.3, -1.2)$ m, and red dot is the odor source location $(0, 5)$ m. (a) Moth-inspired method. Search time: 58 s; travel distance: 8.76 m (b) Adaptive moth-inspired method. Search time: 43 s; travel distance: 6.92 m (c) RL-based method. Search time: 75 s; travel distance: 10.81 m.

are presented in Section 6.3.2. Each proposed method was evaluated in repeat tests to analyze the statistic result.

6.3.1 Adaptive Behavior-based and RL-based Methods

6.3.1.1 Sample Trials

In sample trials, the robot starts from a downwind area in a laminar flow environment, and the odor source is placed at $(0, 5)$ m. Fig. 6.4 shows search trajectories of the traditional moth-inspired method [37], the proposed adaptive behavior-based method, and the proposed RL-based method. It can be observed that all methods can correctly find the odor source location, where the adaptive behavior-based method achieves the shortest search time compared to other methods.



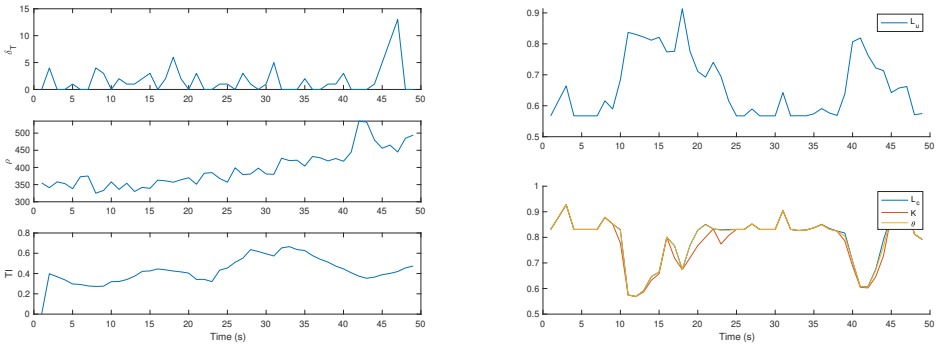
(a) $t = 1s$ (b) $t = 15s$ (c) $t = 30s$

(d) $t = 47s$ (e) $t = 50s$ (f) $t = 58s$

FIGURE 6.5: Snapshots of an OSL test with the moth-inspired method. The robot position is highlighted with the a red rectangle, and the robot correctly finds the odor source at 58 s.

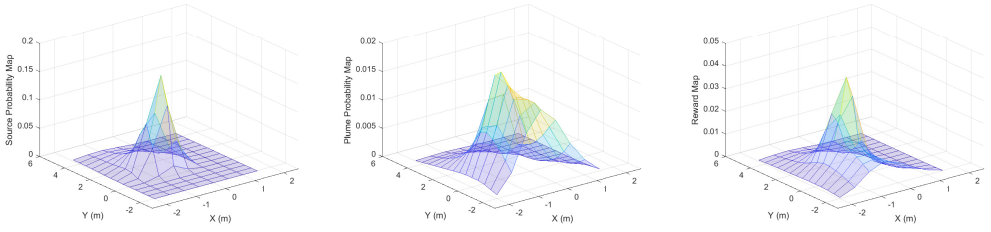
Specifically, Fig. 6.5 presents snapshots of the robot directed by the traditional moth-inspired navigation method. At the beginning of the search (i.e., from $t = 1$ to 15 s), the robot alternates between the 'track-in' and 'track-out' behavior and moves upwind. At $t = 30$ s, the robot loses plume contact and performs the 'reacquire' behavior to search plume over a large area. The robot re-detects plumes at $t = 47$ s and turns back to the 'track-in' behavior at $t = 50$ s. Then, the robot continuously detects plumes and correctly finds the odor source at $t = 58$ s, and the travel distance in this trial is 8.76 m.

Fig. 6.6 shows plots of inputs and outputs of the fuzzy controller in the proposed adaptive behavior-based method. It can be seen that behavior parameters can adaptively change according to different search situations: when the robot is far from the odor source and the estimated airflow characteristic is turbulent (i.e., $t = 5$ s), values of L_c, K, and θ are large to emphasize cross-wind movements to increase the probability of detecting plumes; when the robot is near to the odor source (i.e., the robot continuously detects plumes between $t = 40$ s to $t = 45$ s, indicating the robot moves near to the odor source), values of L_u is increased, improving the upwind movement. The robot finds the



(a) (b)

FIGURE 6.6: Fuzzy inputs and outputs of the implemented fuzzy controller in the proposed adaptive behavior-based method. (a) Fuzzy inputs include plume non-detection period δ_T, odor concentration ρ, and turbulent intensity TI. (b) Fuzzy outputs are coefficients that adjust search behaviors, including L_u (on the top plot), L_c, K, and θ (on the bottom plot).



(a) $t = 20$ s (b) $t = 20$ s (c) $t = 20$ s

(d) $t = 70$ s (e) $t = 70$ s (f) $t = 70$ s

FIGURE 6.7: Results from the source mapping, plume mapping, and fusion algorithms in the proposed reinforcement learning-based method. The first row of diagrams, i.e., (a), (b), and (c), shows the source probability map, plume distribution map, and reward map at $t = 20$ s, respectively. The second row of diagrams, i.e., (d), (e), and (f), shows the aforementioned maps at $t = 70$ s.

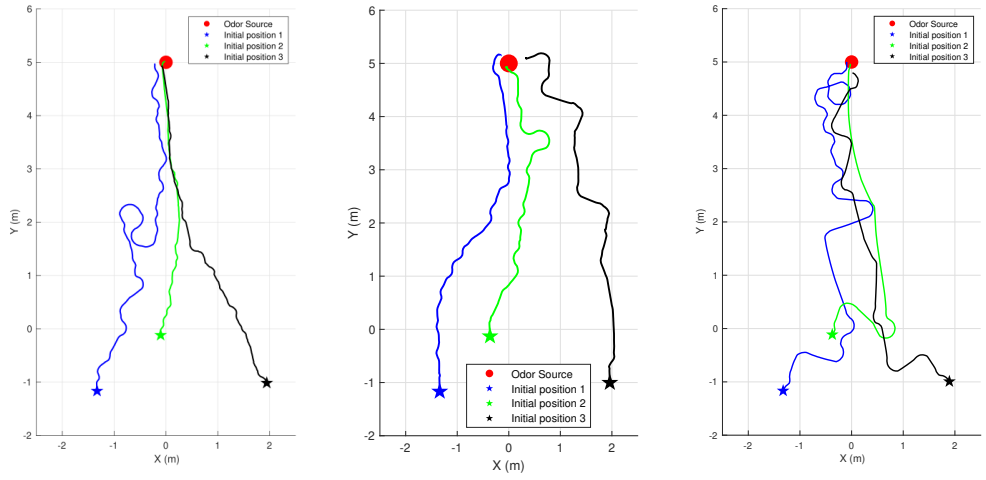
odor source at 43 s and travels 6.92 m.

As for the RL-based method, Fig. 6.7 presents the source probability map, plume distribution map, and the fused reward map at two time steps during the plume tracing, i.e., $t = 20$ s and $t = 70$ s. The size of a sub-region in these maps is 0.5×0.5 m^2, which is

comparable to the size of the mobile robot. When $t = 20$ s, the fusion coefficient (i.e., the output of the fuzzy inference system) is 0.8, which emphasizes the plume distribution map in the reward map. Thus, the robot tends to find plumes to collect more odor source information instead of chasing the source estimate. When $t = 70$ s, the fusion coefficient becomes 0.2, directing the robot to move toward the source estimate $(0, 4.5)$ m (the actual odor source is at $(0, 5)$ m). The robot reaches the odor source location when $t = 75$ s, and the travel distance is 10.81 m.

Compared these three methods, the proposed behavior-based method achieves the best search performance, where the search time and travel distance are shortest as presented in Fig 6.4. Through experiment results, it is observed that the bio-inspired methods (including both the traditional moth-inspired and the proposed adaptive behavior-based methods) outperform the engineering-based method (i.e., the proposed reinforcement learning-based method) in laminar flow environments. This is because in laminar flow environments, odor plumes form a steady and continuous plume trajectory, where the upwind movement in the 'surge' behavior can quickly guide the robot moving toward the odor source by continuously detecting odor plumes. By contrast, the engineering-based method needs to gain information (i.e., airflow measurements and odor concentrations) to accurately estimate possible odor source locations, requiring a longer time to find the odor source compared to moth-inspired methods. In addition, the proposed adaptive behavior-based method is better than the traditional moth-inspired method, indicating the effectiveness of the proposed fuzzy controller, which can adaptively change the scale of robot search trajectories to fit different search situations. Experiment results from repeat tests (i.e., Section 6.3.1.2) endorse this conclusion.

I also tested the search performance of the proposed navigation methods with different robot initial positions, including left, middle, and right positions. Fig. 6.8 shows robot



(a) (b) (c)

FIGURE 6.8: Robot search trajectories staring at different initial positions. (a) Moth-inspired method. (b) Adaptive behavior-based method. (c) Reinforcement learning-based method.

search trajectories with three navigation methods. It can be seen that all search trajectories terminates at the actual odor source location, i.e., no matter how the robot initial position changes, the robot can correctly find the odor source with three navigation methods.

6.3.1.2 Repeat Tests in Different Airflow Environments

In this group of tests, the proposed adaptive behavior-based and RL-based navigation methods were repeatedly performed in both laminar and turbulent airflow environments. Besides, the traditional moth-inspired method was also implemented in repeat tests to compare the search performance of the proposed methods. The robot starts from the same initial position $(-1, 0)$ m and moves in the same speed, i.e., 0.15 m/s.

Table 6.1 presents search time of three navigation methods in the laminar flow environment. It can be observed that the moth-inspired and adaptive behavior-based methods

TABLE 6.1: Search Time of Three Navigation Methods in the Laminar Flow Environment. μ: Mean Search Time; σ: Standard Deviation of Search Time; f: Success Rate

	Moth-inspired Method (s)	Adaptive Behavior-based Method (s)	RL-based Method (s)
Test 1	54	45	63
Test 2	56	42	68
Test 3	45	41	64
Test 4	40	40	63
Test 5	48	41	52
Test 6	42	41	65
Test 7	40	40	55
Test 8	44	46	52
Test 9	42	52	59
Test 10	53	41	54
μ (s)	46.4	**42.9**	59.5
σ	6.0	**3.8**	5.9
f	100%	**100**%	100%

outperform the RL-based method in most tests. This is because the simple 'surge/casting' behavior pattern is effective to quickly bring the robot close to the odor source in the laminar flow environment, while the RL-based method needs more time to correctly compute source and plume estimates based on the sensed airflow history. When the robot loses the plume contact, the 'reacquire' behavior guides the robot to search plumes in nearby areas, and the robot has a high probability to re-detect plumes due to the steady and stable plume distribution in the laminar flow environment. Moreover, the adaptive behavior-based method achieves a better performance than the original moth-inspired counterpart since the implemented fuzzy controller can dynamically adjust behavior parameters to fit different search situations, improving the search efficiency.

Search results in the turbulent flow environment are presented in Table 6.2. Compared three navigation methods, both moth-inspired and adaptive behavior-based methods fail to find the odor source in Test 4, while the RL-based method succeeds in all tests and achieves the shortest averaged search time 59.5 s. In turbulent airflows, the bio-inspired methods (including both the moth-inspired and adaptive behavior-based methods) can

TABLE 6.2: Search Time of Three Navigation Methods in the Turbulent Flow Environment. μ: Mean Search Time; σ: Standard Deviation of Search Time; f: Success Rate

	Moth-inspired Method (s)	Adaptive Behavior-based Method (s)	RL-based Method (s)
Test 1	60	48	55
Test 2	63	93	53
Test 3	71	38	66
Test 4	-	-	55
Test 5	63	103	65
Test 6	66	90	57
Test 7	85	81	90
Test 8	38	63	76
Test 9	116	75	54
Test 10	108	50	63
μ (s)	87.0	84.1	**59.5**
σ	46.0	46.0	**11.8**
f	90%	90%	**100**%

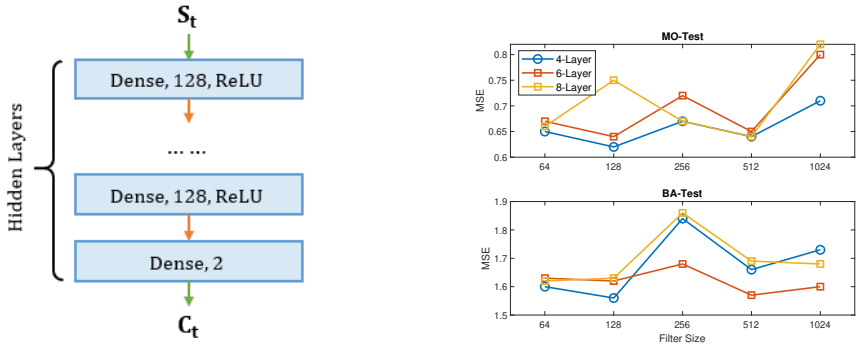
-: Fail to locate the source within 200 s.

hardly keep the robot maintaining inside plumes due to two reasons: 1) plumes are scratched by turbulent airflows to form an intermittent trajectory, deteriorating the performance of the 'surge' and 'casting' behaviors; 2) the decision (i.e., heading command) is made via the instantaneous airflow measurements, which vary significantly and could lead the robot to the wrong directions. By contrast, the RL-based method can calculate source and plume estimates based on the recorded airflow history, which is more reliable than the instantaneous measurements.

6.3.2 DL-based Methods

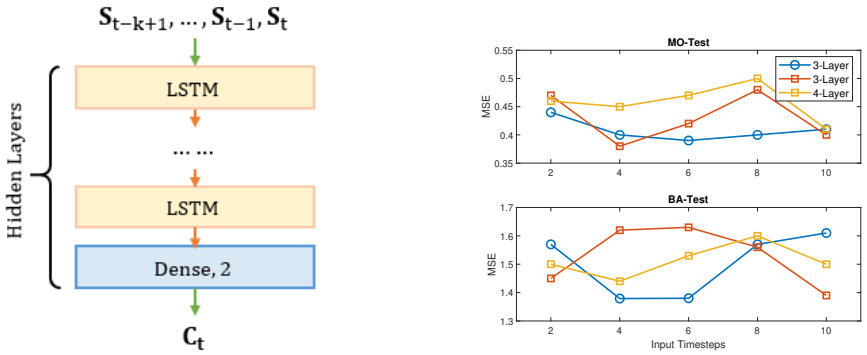
6.3.2.1 Develop DNNs for DL-based Methods

I re-design DNN structures for fitting the training data collected from on-vehicle implementations, including a feedforward (FNN) and long short-term memory (LSTM) networks. The motivation for choosing FNN is that I want to use a simple DNN structure to investigate the viability of implementing DL approaches on OSL problems. Besides,



(a) (b)

FIGURE 6.9: (a) The structure of the proposed FNNs. Notations inside a blue layer represent the layer type, filter size, and activation function, respectively. A dense layer indicates a fully-connected neural network; (b) MSEs of FNNs with varying filter sizes and hidden layers on two testing data sets, including MO-test and BA-test.



(a) (b)

FIGURE 6.10: (a) The structure of the proposed LSTM network. Inputs of the LSTM network include previous sensor readings, i.e., $\mathbf{S}_{t-k+1}$ to $\mathbf{S}_t$, where the length of inputs is k. Multiple LSTM cells are stacked to form a deep LSTM network; (b) MSEs of LSTM networks with the varying time steps of inputs (i.e., k) on testing data sets.

the intuitive FNN could also be employed as the baseline to evaluate the performance of other types of DNN models in the OSL problem. To determine the optimal structure of the FNN network, varying numbers of hidden layers and filter sizes are evaluated with the testing data sets. Fig. 6.9 shows the mean square errors (MSEs) of implementing different FNNs with varying hidden layers on testing data sets. It can be observed that larger models (i.e., more layers) achieve better performances (i.e., lower MSEs) but overfit (i.e., the MSE increases) when the model is too complicated. Based on plots, the FNN with 4-layer and 128 filters is selected for training both MO-train and BA-train.

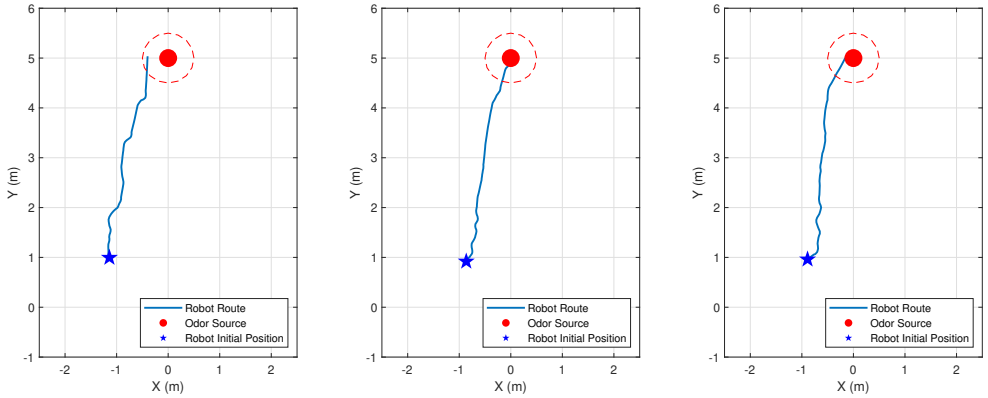
Besides the simple FNN network, I also train a LSTM network. Unlike standard FNNs, LSTMs have feedback connections, which bring the previous output to the input at the current time step. In this work, inputs of a neural network are onboard sensor readings and outputs are robot commands. By using LSTMs, the sensor data history can be involved in the calculation of robot commands at the current time step. This feature may be useful in learning the Bayesian-inference method, which deduces possible odor source locations via sensed airflow history. Similar to FNNs, we determine the optimal structure of LSTM by examining multiple combinations of layer numbers and the size of input (i.e., how many time steps k are included in the input). Fig. 6.10 shows the plot of MSEs of implementing different structure LSTMs on testing data sets. Based on the plot, we choose the input time step as 4 for both MO-Train and BA-Train, where the number of LSTM layers is 3 for the MO-Train and 2 for the BA-Train.

After training with the training data sets generated by two expert methods, i.e., MO-Train and BA-Train, four DNNs are obtained, termed FNN-Moth, LSTM-Moth, FNN-BA, and LSTM-BA. This notation consists of the type of DNN and the corresponding training data set, e.g., FNN-Moth is the FNN trained with MO-Train.

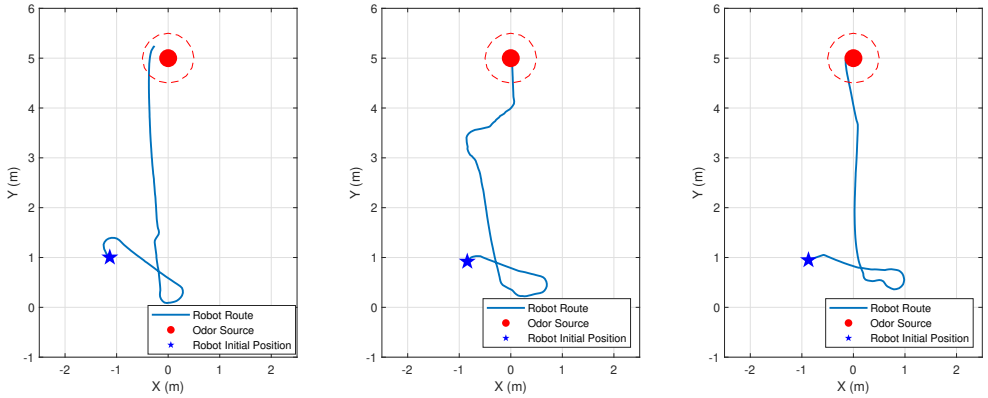
6.3.2.2 Search Results in Seen Environments

I first demonstrate the search results generated by implementing the trained DNNs in a seen environment, where the odor source location is placed at $(0, 5)$ m and the airflow direction points to the negative side of the y axis.

Fig. 6.11 presents search trajectories generated by expert methods and DNNs. Fig. 6.11(a), 6.11(b), and 6.11(c) show search trajectories generated by the original moth-inspired method, FNN-Moth, and LSTM-Moth, respectively. It can be seen that three



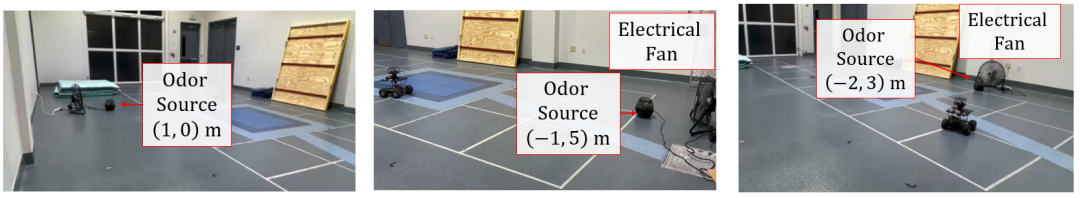
(a) Moth-inspired (b) FNN-Moth (c) LSTM-Moth



(d) Bayesian-based (e) FNN-BA (f) LSTM-BA

FIGURE 6.11: Six sample OSL trials with different navigation methods.

search trajectories are very similar and all end up with the odor source location, indicating that the proposed DNNs can mimic the moth-inspired method to correctly find the odor source. As for the Bayesian-inference method (presented in Fig. 6.11(d)), the search trajectory produced by the LSTM-BA (i.e., Fig. 6.11(f)) achieves a shorter search time than the FNN-BA (i.e., Fig. 6.11(e)), indicating a better search performance. This is because the context understanding ability of the LSTM network is more suitable for learning the Bayesian-inference method, which requires the knowledge of the sensor reading history to estimate possible odor source locations. It can be observed that search trajectories generated by DNNs are similar to the expert methods, and both FNN and LSTM can find the odor source location in these sample OSL trials. However,



(a) (b) (c)

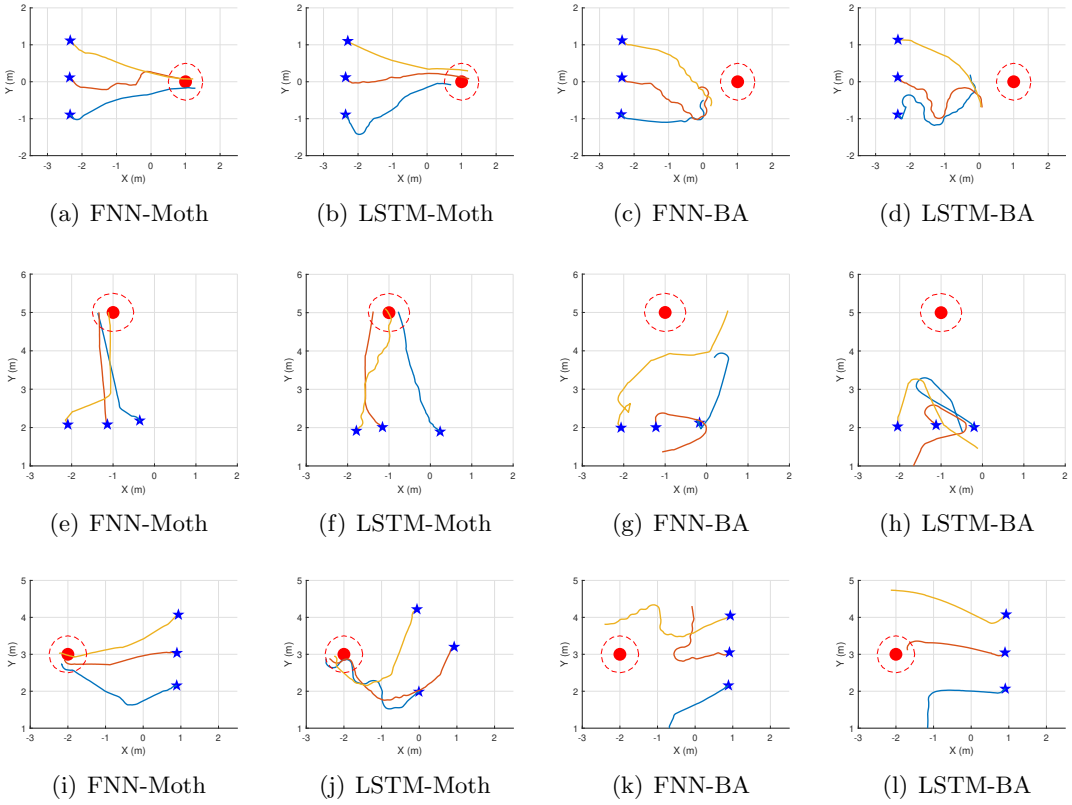
FIGURE 6.12: Unseen search environment where odor source locations are located at (a) $(1, 0)$ m, (b) $(-1, 5)$ m, and (c) $(-2, 3)$ m. New airflow directions are created in unseen environments, where the airflow direction points to (a) negative side of x axis, (b) negative side of y axis, and (c) positive side of x axis.

it is possible that DNNs memorize the final odor source location instead of learning the actual searching strategies demonstrated by the expert methods. Thus, trained DNNs need to be examined in previously unseen environments to verify their validities.

6.3.2.3 Search Results in Unseen Environments

In this group of tests, trained DNNs are tested in unseen environments. As shown in Fig. 6.12, we choose three new odor source locations and airflow directions to construct the testing environment, and for each new odor source location, various OSL trials were conducted with different robot initial positions.

Fig. 6.13 shows the search results of trained DNNs in unseen environments. It can be observed that both FNN-Moth and LSTM-Moth networks, trained by the moth-inspired method, can correctly find the odor source in new environments. In these trials, the robot demonstrates the ability to mimic the moth-inspired method, which can effectively find the odor source by performing the upwind movement (similar to the 'surge' behavior) or the crosswind excursion (like the 'casting' behavior) to move toward the odor source. By contrast, the performance of FNN-BA and LSTM-BA, trained with the Bayesian-inference method, is not satisfied, where only a few of tests successfully find the odor source location.



(a) FNN-Moth (b) LSTM-Moth (c) FNN-BA (d) LSTM-BA

(e) FNN-Moth (f) LSTM-Moth (g) FNN-BA (h) LSTM-BA

(i) FNN-Moth (j) LSTM-Moth (k) FNN-BA (l) LSTM-BA

FIGURE 6.13: Robot search trajectories generated by the proposed DNNs with different robot initial positions in previous unseen environments. In these diagrams, blue stars indicate robot initial positions, the red dot represents the odor source location, and the red dotted line shows the source localization range (i.e., 0.5 m): the odor source is considered as located if the robot is inside this range.

This result is anticipated since the search strategy in the moth-inspired method is much simpler than the one demonstrated by the Bayesian-inference method, which finds the odor source relying on mathematical calculations. For the Bayesian-inference method, the underlying calculation process cannot directly reflect on the robot search behaviors, making it difficult for DNNs to mimic its search strategy. Given the same amount of training data sets, DNNs are more likely to learn the simple moth-inspired method than the Bayesian-inference counterpart. One possible solution is adding more demonstrations with different odor source locations to diversify the training data set. Still, the increasing time and efforts to collect training data and raise training time are also significant.

6.3.2.4 Separate the Locations of Odor Source and Electrical Fan

In previous tests, the odor source was always placed with the electrical fan. A reasonable assumption is that the robot is attracted by the wind direction instead of odor concentration during the search. To eliminate the assumption that the robot was attracted solely by the wind direction, two groups of tests were conducted.

The odor source was removed in the first group of tests, and only an electrical fan was placed inside the search area. The implemented deep learning model is LSTM neural network. Fig. 6.14 shows snapshots of a test in group one. As we can see, robot did not go to the fan's location. This test was repeated three times, and the robot did not go to the fan's location in all tests.

(a) (b) (c) (d)

FIGURE 6.14: Snapshots of a test where the odor source is not inside the search area. The robot did not go to the electrical fan at the end of the search.

(a) (b) (c) (d)

FIGURE 6.15: Snapshots of a test where the odor source is separate with the electrical fan. The robot finds the odor source location even though the electrical fan is placed differently with the odor source.

In the second group of tests, the odor source was placed differently from the electrical fan. Fig. 6.15 presents snapshots of a tests in group two. We can see that the robot can correctly find the odor source location even though the electrical fan was at a different

TABLE 6.3: Search Results of Four Navigation Methods in Repeated Tests. μ: Mean Search Time; σ: Standard Deviation

Test Name	Moth-inspired Method (s)	Bayesian-inference Method (s)	FNN-Moth (s)	LSTM-Moth (s)
Test 1	18	40	21	20
Test 2	21	24	19	20
Test 3	19	23	20	21
Test 4	20	22	20	21
Test 5	19	23	20	21
Test 6	26	28	25	30
Test 7	27	28	28	28
Test 8	28	42	25	26
Test 9	25	30	30	29
Test 10	27	27	27	30
μ (s)	**23.0**	28.7	**23.5**	24.6
σ	**3.9**	7.0	**4.0**	4.4

location. Similar to the group one tests, this test was repeated three times, and the robot always found the odor source.

Through these tests, I found that the wind direction does not solely attract the robot during the search. For the proposed deep learning-based method, detecting odor plumes plays a vital role in finding the odor source location.

6.3.2.5 Compare with Expert Methods in Repeated Tests

In this group of tests, we implement the FNN-Moth and LSTM-Moth networks in an unseen environment, where the odor source is located at $(-1, 5)$ m. To evaluate the search performance, two DNNs are compared with two expert methods. For each navigation method, 10 OSL trials were conducted. The robot initial position is at $(-0.7, 2)$ m in Test 1-5 and changes to a far position at $(-1.2, 1)$ m in Test 6-10. Search results are reported in Table 6.3.

It can be observed in Table 6.3 that all navigation methods can correctly navigate the robot to find the odor source. In terms of the averaged search time (μ) and standard deviation (σ), the moth-inspired method achieves the best performance among other

methods. Two DNNs attain a comparable search performance with the moth-inspired method. Moreover, the FNN-Moth is slightly better than the LSTM-Moth network due to the lower averaged search time (23.5 s v.s. 24.6 s) and standard deviation (4.0 v.s. 4.4). Repeated tests reflect that both neural networks can effectively learn the moth-inspired method to locate an odor source in a new search environment.

6.4 Summary of this Chapter

This chapter presents the experimental results of the proposed olfactory-based navigation methods, namely adaptive behavior-based, RL-based, and DL-based methods. A mobile robot is constructed as the robotic platform to implement different navigation methods. For adaptive behavior-based and RL-based methods, experiment results show that in terms of the averaged search time and success rate, the adaptive moth-inspired method has the best search performance in the laminar flow environment, whereas the RL-based method outperforms the others in the turbulent flow environment.

For the DL-based plume tracing algorithms, two DNNs are devised, including FNN and LSTM networks. DNNs calculate robot commands that navigate the robot to find the odor source based on onboard sensor readings during the plume tracing process. Two expert methods, namely moth-inspired and Bayesian-inference methods, are employed as expert methods to generated training data sets. Hundreds of OSL trials were conducted to collect training data, and after training, DNNs are validated in unseen search environments. Experiment results show that both FNN and LSTM can mimic the moth-inspired method but cannot effectively learn the Bayesian-inference method given the same amount of training data. In repeated tests, the search performance of DNNs

is comparable with the moth-inspired method and better than the Bayesian-inference method in terms of the averaged search time and standard deviation.

Chapter 7

Conclusion and Future Works

This chapter presents a summary of this dissertation, the conclusions of the research work, and future research directions.

7.1 Conclusions

This dissertation focuses on robotic odor source localization (OSL), which designs olfactory-based navigation algorithms to guide a mobile robot in finding an odor source in unknown environments. Three types of algorithms are presented, including adaptive behavior-based, reinforcement learning-based (RL-based), and deep learning-based (DL-based) algorithms. The objective is to improve the search efficiency during the plume tracing process while maintaining a low computational cost.

Specifically, the adaptive behavior-based method is a modified version of the traditional moth-inspired method. By integrating a customized fuzzy inference system, this method enables the robot to dynamically adjust its search behaviors and the scale of trajectories depending on the current search situation. As a result, the robot can increase the scale

of search trajectories when plumes are absent to increase the probability of re-detecting plumes and decrease the scale when the robot is near the source. Test results from both simulation and on-vehicle tests show that the proposed adaptive behavior-based method improves the search time compared to the conventional bio-inspired method. In addition, the computational cost is significantly reduced compared to the conventional engineering-based method (e.g., Bayesian-inference method), which requires calculating the cell-based probability for all cells inside the search area in every iteration loop.

The reinforcement learning-based algorithm concentrates on solving the OSL problem via an engineering-based approach. In this method, source and plume estimations, calculated by the partially observable Markov decision process-based (POMDP-based) source mapping and hidden Markov model-based (HMM-based) plume mapping algorithms, are dynamically fused via a fuzzy controller. This fuzzy controller estimates the distance between the robot and the odor source: when these two objects are close, the source estimation is dominant; otherwise, the plume estimations overweight the source estimation in the fused result. Simulation results show that the proposed reinforcement learning-based method outperforms the traditional moth-inspired and engineering-based methods in search time and success rate. Results from on-vehicle tests verify this conclusion.

The principle idea of the deep learning-based methods is to leverage the odor source localization problem without explicating specific plume tracing algorithms. The goal is to design a machine learning (ML) model that can learn an effective navigation strategy to find an odor source by watching other successful olfactory-based navigation algorithms. Two ML models are trained, including an adaptive neural fuzzy inference system (ANFIS) and deep neural networks (DNNs). Traditional moth-inspired and engineering-based methods were selected as expert methods to generate training data for supervised learning. After training, simulation results show that the proposed ANFIS and DNNs

can correctly find the odor source in previously unseen environments. Compared to traditional methods, the ML models trained by the fused training data set outperform them in search time and success rate.

7.2 Future Research

With the rapid development of robotics and autonomous systems, integrating advanced AI techniques in robot planning, navigation, and controls enlightens a promising research direction. For future research, I will continue developing AI-based algorithms to study research problems in robotics and autonomous systems and implement these algorithms in real-world applications. I summarize my future research in three directions:

- **Advanced Olfactory-based Navigation Methods.** With my expertise in robotic olfaction, I would like to develop more advanced olfactory-based navigation algorithms using AI techniques. More complicated situations will be considered, such as locating multiple odor sources, searching in complex terrains and unstructured environments, searching with multiple robots in the collaboration, etc. I will utilize advanced AI techniques to design these navigation algorithms, including deep reinforcement learning, multi-agent reinforcement learning, transfer learning, etc. I would also like to research integrating other robotic sensing abilities (e.g., computer vision) to improve search efficiency in finding the source. Besides, I strive to apply the designed algorithms to solve challenging real-world problems, such as locating gas leaks, monitoring air pollution, finding wildfire locations, etc.

- **Intelligent Decision-making Algorithms in Robotics and Autonomous Systems.** The core of my olfactory-based navigation algorithms is to develop a methodology that makes intelligent decisions to guide the robot in the dynamic environment.

This principle can be applied more broadly in searching and data collection with sporadic cues and partial information. In the future, I would like to research utilizing and adapting the principle of my algorithms on other autonomous systems, such as autonomous driving vehicles. I will concentrate on developing intelligent decision-making algorithms in the planning and navigation procedures. Specifically, I will start with small-scale projects, such as developing autonomous delivery robots, to verify algorithms in a safe and easily controlled environment. Then, I will use preliminary outcomes to direct the research on more complicated autonomous systems, e.g., self-driving techniques. These projects will also provide graduate and undergraduate students with opportunities to gain research and hands-on experiences.

- **AI-based Multi-agent Coordination Algorithms.** Over the past decade, multi-agent robotic systems have gained significant research momentum for their high efficiency in solving real-world problems. Besides, with the recent developments in AI, we have witnessed an increasing number of multi-agent applications using AI methods to improve system performance. Based on my previous research experience in AI-based multi-agent coordination algorithms, I plan to integrate more advanced AI techniques in solving multi-agent problems, like multi-agent path planning, behavior coordination, formation control, and performance improvement. In addition, I will research implementing the designed algorithms in real-world environments, starting from small-scale projects (e.g., coordination of a fleet of mobile robots). I will use the research experiences and experiment results to improve the designed algorithms and eventually implement them in practical applications to solve real-world problems (e.g., search-and-rescue tasks).

Bibliography

[1] U. Montanaro, S. Dixit, S. Fallah, M. Dianati, A. Stevens, D. Oxtoby, and A. Mouzakitis, "Towards connected autonomous driving: review of use-cases," *Vehicle system dynamics*, vol. 57, no. 6, pp. 779–814, 2019.

[2] G. Kowadlo and R. A. Russell, "Robot odor localization: a taxonomy and survey," *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 869–894, 2008.

[3] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[4] S. Soldan, G. Bonow, and A. Kroll, "Robogasinspector-a mobile robotic system for remote leak sensing and localization in large industrial environments: Overview and first results," *IFAC Proceedings Volumes*, vol. 45, no. 8, pp. 33–38, 2012.

[5] R. A. Russell, "Robotic location of underground chemical sources," *Robotica*, vol. 22, no. 1, pp. 109–115, 2004.

[6] B. L. Villarreal and J. L. Gordillo, "Directional aptitude analysis in odor source localization techniques for rescue robots applications," in *2011 10th Mexican International Conference on Artificial Intelligence*, pp. 109–114, IEEE, 2011.

[7] G. Ferri, M. V. Jakuba, and D. R. Yoerger, "A novel method for hydrothermal vents prospecting using an autonomous underwater robot," in *2008 IEEE International Conference on Robotics and Automation*, pp. 1055–1060, IEEE, 2008.

[8] J. A. Farrell, J. Murlis, X. Long, W. Li, and R. T. Cardé, "Filament-based atmospheric dispersion model to achieve short time-scale structure of odor plumes," *Environmental fluid mechanics*, vol. 2, no. 1-2, pp. 143–169, 2002.

[9] W. Naeem, R. Sutton, and J. Chudley, "Chemical plume tracing and odour source localisation by autonomous vehicles," *The Journal of Navigation*, vol. 60, no. 2, pp. 173–190, 2007.

[10] H. Ishida, K.-i. Suetsugu, T. Nakamoto, and T. Moriizumi, "Study of autonomous mobile sensing system for localization of odor source using gas sensors and anemometric sensors," *Sensors and Actuators A: Physical*, vol. 45, no. 2, pp. 153–157, 1994.

[11] X.-x. Chen and J. Huang, "Odor source localization algorithms on mobile robots: A review and future outlook," *Robotics and Autonomous Systems*, vol. 112, pp. 123–136, 2019.

[12] R. T. Cardé and A. Mafra-Neto, "Mechanisms of flight of male moths to pheromone," in *Insect pheromone research*, pp. 275–290, Springer, 1997.

[13] L. L. López, V. Vouloutsi, A. E. Chimeno, E. Marcos, S. B. i Badia, Z. Mathews, P. F. Verschure, A. Ziyatdinov, and A. P. i Lluna, "Moth-like chemo-source localization and classification on an indoor autonomous robot," in *On Biomimetics*, IntechOpen, 2011.

[14] S. Pang and F. Zhu, "Reactive planning for olfactory-based mobile robots," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4375–4380, IEEE, 2009.

[15] L. Wang, S. Pang, and J. Li, "Olfactory-based navigation via model-based reinforcement learning and fuzzy inference methods," *IEEE Transactions on Fuzzy Systems*, 2020.

[16] J. A. Farrell, S. Pang, and W. Li, "Plume mapping via hidden Markov methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, no. 6, pp. 850–863, 2003.

[17] J. Adler, "The sensing of chemicals by bacteria," *Scientific American*, vol. 234, no. 4, pp. 40–47, 1976.

[18] L. Marques, U. Nunes, and A. T. de Almeida, "Olfaction-based mobile robot navigation," *Thin solid films*, vol. 418, no. 1, pp. 51–58, 2002.

[19] G. Sandini, G. Lucarini, and M. Varoli, "Gradient driven self-organizing systems," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, vol. 1, pp. 429–432, IEEE, 1993.

[20] F. W. Grasso, T. R. Consi, D. C. Mountain, and J. Atema, "Biomimetic robot lobster performs chemo-orientation in turbulence using a pair of spatially separated sensors: Progress and challenges," *Robotics and Autonomous Systems*, vol. 30, no. 1-2, pp. 115–131, 2000.

[21] R. A. Russell, A. Bab-Hadiashar, R. L. Shepherd, and G. G. Wallace, "A comparison of reactive robot chemotaxis algorithms," *Robotics and Autonomous Systems*, vol. 45, no. 2, pp. 83–97, 2003.

[22] A. Lilienthal and T. Duckett, "Experimental analysis of gas-sensitive Braitenberg vehicles," *Advanced Robotics*, vol. 18, no. 8, pp. 817–834, 2004.

[23] H. Ishida, G. Nakayama, T. Nakamoto, and T. Moriizumi, "Controlling a gas/odor plume-tracking robot based on transient responses of gas sensors," *IEEE Sensors Journal*, vol. 5, no. 3, pp. 537–545, 2005.

[24] J. Murlis and C. Jones, "Fine-scale structure of odour plumes in relation to insect orientation to distant pheromone and other attractant sources," *Physiological Entomology*, vol. 6, no. 1, pp. 71–86, 1981.

[25] J. S. Kennedy and D. Marsh, "Pheromone-regulated anemotaxis in flying moths," *Science*, vol. 184, no. 4140, pp. 999–1001, 1974.

[26] R. T. Cardé and M. A. Willis, "Navigational strategies used by insects to find distant, wind-borne sources of odor," *Journal of chemical ecology*, vol. 34, no. 7, pp. 854–866, 2008.

[27] J. Elkinton, C. Schal, T. Onot, and R. Cardé, "Pheromone puff trajectory and upwind flight of male gypsy moths in a forest," *Physiological Entomology*, vol. 12, no. 4, pp. 399–406, 1987.

[28] H. Ishida, Y. Wada, and H. Matsukura, "Chemical sensing in robotic applications: A review," *IEEE Sensors Journal*, vol. 12, no. 11, pp. 3163–3173, 2012.

[29] R. Kanzaki, N. Sugi, and T. Shibuya, "Self-generated zigzag turning of Bombyx mori males during pheromone-mediated upwind walking (Physology)," *Zoological science*, vol. 9, no. 3, pp. 515–527, 1992.

[30] Y. Kuwana and I. Shimoyama, "A pheromone-guided mobile robot that behaves like a silkworm moth with living antennae as pheromone sensors," *The International Journal of Robotics Research*, vol. 17, no. 9, pp. 924–933, 1998.

[31] G. De Croon, L. O'connor, C. Nicol, and D. Izzo, "Evolutionary robotics approach to odor source localization," *Neurocomputing*, vol. 121, pp. 481–497, 2013.

[32] T. Lochmatter, X. Raemy, L. Matthey, S. Indra, and A. Martinoli, "A comparison of casting and spiraling algorithms for odor source localization in laminar flow," in *2008 IEEE International Conference on Robotics and Automation*, pp. 1138–1143, IEEE, 2008.

[33] W. Li, J. A. Farrell, S. Pang, and R. M. Arrieta, "Moth-inspired chemical plume tracing on an autonomous underwater vehicle," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 292–307, 2006.

[34] W. Li, "Abstraction of odor source declaration algorithm from moth-inspired plume tracing strategies," in *2006 IEEE International Conference on Robotics and Biomimetics*, pp. 1024–1029, IEEE, 2006.

[35] S. Pang, "Development of a guidance system for AUV chemical plume tracing," in *OCEANS 2006*, pp. 1–6, IEEE, 2006.

[36] B. Luo, Q.-H. Meng, J.-Y. Wang, and M. Zeng, "A flying odor compass to autonomously locate the gas source," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 1, pp. 137–149, 2017.

[37] J. A. Farrell, S. Pang, and W. Li, "Chemical plume tracing via an autonomous underwater vehicle," *IEEE Journal of Oceanic Engineering*, vol. 30, no. 2, pp. 428–442, 2005.

[38] S. Shigaki, T. Sakurai, N. Ando, D. Kurabayashi, and R. Kanzaki, "Time-varying moth-inspired algorithm for chemical plume tracing in turbulent environment," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 76–83, 2017.

[39] S. Shigaki, Y. Shiota, D. Kurabayashi, and R. Kanzaki, "Modeling of the adaptive chemical plume tracing algorithm of an insect using fuzzy inference," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 1, pp. 72–84, 2019.

[40] S. Shigaki, S. Haigo, C. H. Reyes, T. Sakurai, R. Kanzaki, D. Kurabayashi, and H. Sezutsu, "Analysis of the role of wind information for efficient chemical plume tracing based on optogenetic silkworm moth behavior," *Bioinspiration & biomimetics*, vol. 14, no. 4, p. 046006, 2019.

[41] A. Liberzon, K. Harrington, N. Daniel, R. Gurka, A. Harari, and G. Zilman, "Moth-inspired navigation algorithm in a turbulent odor plume from a pulsating source," *PloS one*, vol. 13, no. 6, p. e0198422, 2018.

[42] G. Ferri, E. Caselli, V. Mattoli, A. Mondini, B. Mazzolai, and P. Dario, "Spiral: A novel biologically-inspired algorithm for gas/odor source localization in an indoor environment with no strong airflow," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 393–402, 2009.

[43] F. Rahbar, A. Marjovi, P. Kibleur, and A. Martinoli, "A 3-d bio-inspired odor source localization and its validation in realistic environmental conditions," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3983–3989, IEEE, 2017.

[44] P. Pyk, S. B. i Badia, U. Bernardet, P. Knüsel, M. Carlsson, J. Gu, E. Chanie, B. S. Hansson, T. C. Pearce, and P. F. Verschure, "An artificial moth: Chemical source

localization using a robot based neuronal model of moth optomotor anemotactic search," *Autonomous Robots*, vol. 20, no. 3, pp. 197–213, 2006.

[45] P. B. Reeder and B. W. Ache, "Chemotaxis in the florida spiny lobster, panulirus argus," *Animal Behaviour*, vol. 28, no. 3, pp. 831–839, 1980.

[46] T. Consi, J. Atema, C. Goudey, J. Cho, and C. Chryssostomidis, "Auv guidance with chemical signals," in *Proceedings of IEEE Symposium on Autonomous Underwater Vehicle Technology (AUV'94)*, pp. 450–455, IEEE, 1994.

[47] F. W. Grasso, J. A. Basil, and J. Atema, "Toward the convergence: robot and lobster perspectives of tracking odors to their source in the turbulent marine environment," in *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC) held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) Intell*, pp. 259–264, IEEE, 1998.

[48] T. Lochmatter and A. Martinoli, "Tracking odor plumes in a laminar wind field with bio-inspired algorithms," in *Experimental robotics*, pp. 473–482, Springer, 2009.

[49] V. Hernandez Bennetts, A. J. Lilienthal, P. Neumann, and M. Trincavelli, "Mobile robots for localizing gas emission sources on landfill sites: is bio-inspiration the way to go?," *Frontiers in neuroengineering*, vol. 4, p. 20, 2012.

[50] T. Lochmatter, "Bio-inspired and probabilistic algorithms for distributed odor source localization using mobile robots," 2010.

[51] S. Pang and J. A. Farrell, "Chemical plume source localization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 5, pp. 1068–1080, 2006.

[52] J.-G. Li, Q.-H. Meng, Y. Wang, and M. Zeng, "Odor source localization using a mobile robot in outdoor airflow environments with a particle filter algorithm," *Autonomous Robots*, vol. 30, no. 3, pp. 281–292, 2011.

[53] X. Chen and J. Huang, "Combining particle filter algorithm with bio-inspired anemotaxis behavior: A smoke plume tracking method and its robotic experiment validation," *Measurement*, vol. 154, p. 107482, 2020.

[54] H. Jiu, Y. Chen, W. Deng, and S. Pang, "Underwater chemical plume tracing based on partially observable markov decision process," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419831874, 2019.

[55] Z. A. Saigol, R. W. Dearden, J. L. Wyatt, and B. J. Murton, "Information-lookahead planning for auv mapping," in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[56] H. Hu, S. Song, and C. P. Chen, "Plume tracing via model-free reinforcement learning method," *IEEE transactions on neural networks and learning systems*, 2019.

[57] M. Vergassola, E. Villermaux, and B. I. Shraiman, "'infotaxis' as a strategy for searching without gradients," *Nature*, vol. 445, no. 7126, p. 406, 2007.

[58] H.-f. Jiu, S. Pang, J.-l. Li, and B. Han, "Odor plume source localization with a pioneer 3 mobile robot in an indoor airflow environment," in *IEEE SOUTHEASTCON 2014*, pp. 1–6, IEEE, 2014.

[59] J.-G. Li, M.-L. Cao, and Q.-H. Meng, "Chemical Source Searching by Controlling a Wheeled Mobile Robot to Follow an Online Planned Route in Outdoor Field Environments," *Sensors*, vol. 19, no. 2, p. 426, 2019.

[60] Y. Q. Chen and Z. Wang, "Formation control: a review and a new consideration," in *2005 IEEE/RSJ International conference on intelligent robots and systems*, pp. 3181–3186, IEEE, 2005.

[61] T. Lochmatter, E. A. Göl, I. Navarro, and A. Martinoli, "A plume tracking algorithm based on crosswind formations," in *Distributed Autonomous Robotic Systems*, pp. 91–102, Springer, 2013.

[62] J. M. Soares, A. P. Aguiar, A. M. Pascoal, and A. Martinoli, "A graph-based formation algorithm for odor plume tracing," in *Distributed Autonomous Robotic Systems*, pp. 255–269, Springer, 2016.

[63] A. Marjovi and L. Marques, "Optimal swarm formation for odor plume finding," *IEEE transactions on cybernetics*, vol. 44, no. 12, pp. 2302–2315, 2014.

[64] J. M. Soares, A. Marjovi, J. Giezendanner, A. Kodiyan, A. P. Aguiar, A. M. Pascoal, and A. Martinoli, "Towards 3-d distributed odor source localization: an extended graph-based formation control algorithm for plume tracking," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1729–1736, IEEE, 2016.

[65] Y. Tan and Z. Zheng, "Research advance in swarm robotics," *Defence Technology*, vol. 9, no. 1, pp. 18–39, 2013.

[66] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Distributed odor source localization," *IEEE Sensors Journal*, vol. 2, no. 3, pp. 260–271, 2002.

[67] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[68] L. Marques, U. Nunes, and A. T. de Almeida, "Particle swarm-based olfactory guided search," *Autonomous Robots*, vol. 20, no. 3, pp. 277–287, 2006.

[69] Z. Fu, Y. Chen, Y. Ding, and D. He, "Pollution source localization based on multi-uav cooperative communication," *IEEE Access*, vol. 7, pp. 29304–29312, 2019.

[70] Q.-H. Meng, W.-X. Yang, Y. Wang, and M. Zeng, "Collective odor source estimation and search in time-variant airflow environments using mobile robots," *Sensors*, vol. 11, no. 11, pp. 10415–10443, 2011.

[71] Q. Lu and P. Luo, "A learning particle swarm optimization algorithm for odor source localization," *International Journal of Automation and Computing*, vol. 8, no. 3, pp. 371–380, 2011.

[72] X. Huang, "Improved 'infotaxis' algorithm-based cooperative multi-usv pollution source search approach in lake water environment," *Symmetry*, vol. 12, no. 4, p. 549, 2020.

[73] W. Jatmiko, K. Sekiyama, and T. Fukuda, "A mobile robots pso-based for odor source localization in dynamic advection-diffusion environment," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4527–4532, IEEE, 2006.

[74] G. Ferri, E. Caselli, V. Mattoli, A. Mondini, B. Mazzolai, and P. Dario, "Exploration particle swarm optimization method for gas/odor source localization in an indoor environment with no strong airflow," in *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 841–846, IEEE, 2007.

[75] Y. Yan, R. Zhang, J. Wang, and J. Li, "Modified pso algorithms with "request and reset" for leak source localization using multiple robots," *Neurocomputing*, vol. 292, pp. 82–90, 2018.

[76] Q. Feng, H. Cai, F. Li, X. Liu, S. Liu, and J. Xu, "An improved particle swarm optimization method for locating time-varying indoor particle sources," *Building and environment*, vol. 147, pp. 146–157, 2019.

[77] Q. Feng, H. Cai, Z. Chen, Y. Yang, J. Lu, F. Li, J. Xu, and X. Li, "Experimental study on a comprehensive particle swarm optimization method for locating contaminant sources in dynamic indoor environments with mechanical ventilation," *Energy and buildings*, vol. 196, pp. 145–156, 2019.

[78] T. Lochmatter and A. Martinoli, "Understanding the potential impact of multiple robots in odor source localization," in *Distributed Autonomous Robotic Systems 8*, pp. 239–250, Springer, 2009.

[79] M. Sabelis and P. Schippers, "Variable wind directions and anemotactic strategies of searching for an odour plume," *Oecologia*, vol. 63, no. 2, pp. 225–228, 1984.

[80] W. Li, J. A. Farrell, and R. T. Card, "Tracking of fluid-advected odor plumes: strategies inspired by insect orientation to pheromone," *Adaptive Behavior*, vol. 9, no. 3-4, pp. 143–170, 2001.

[81] L. Kuenen and R. T. Carde, "Strategies for recontacting a lost pheromone plume: casting and upwind flight in the male gypsy moth," *Physiological Entomology*, vol. 19, no. 1, pp. 15–29, 1994.

[82] J. Murlis, J. S. Elkinton, and R. T. Carde, "Odor plumes and how insects use them," *Annual review of entomology*, vol. 37, no. 1, pp. 505–532, 1992.

[83] G. C. Sousa and B. K. Bose, "A fuzzy set theory based control of a phase-controlled converter dc machine drive," *IEEE Transactions on industry Applications*, vol. 30, no. 1, pp. 34–44, 1994.

[84] M. Pratama, J. Lu, and G. Zhang, "Evolving type-2 fuzzy classifier," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 3, pp. 574–589, 2015.

[85] N. Carpman, "Turbulence intensity in complex environments and its influence on small wind turbines," 2011.

[86] F. Russo and N. T. Basse, "Scaling of turbulence intensity for low-speed flow in smooth pipes," *Flow Measurement and Instrumentation*, vol. 52, pp. 101–114, 2016.

[87] J. P. Crimaldi, M. B. Wiley, and J. R. Koseff, "The relationship between mean and instantaneous structure in turbulent passive scalar plumes," *Journal of Turbulence*, vol. 3, no. 14, pp. 1–24, 2002.

[88] J. Elkinton, R. Cardé, and C. Mason, "Evaluation of time-average dispersion models for estimating pheromone concentration in a deciduous forest," *Journal of chemical ecology*, vol. 10, no. 7, pp. 1081–1108, 1984.

[89] B. K. Bose, "Expert system, fuzzy logic, and neural network applications in power electronics and motion control," *Proceedings of the IEEE*, vol. 82, no. 8, pp. 1303–1323, 1994.

[90] J. A. Farrell, S. Pang, W. Li, and R. Arrieta, "Chemical plume tracing experimental results with a remus auv," in *Oceans 2003. Celebrating the Past... Teaming Toward the Future (IEEE Cat. No. 03CH37492)*, vol. 2, pp. 962–968, IEEE, 2003.

[91] W. Jatmiko, K. Sekiyama, and T. Fukuda, "A pso-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment: theory, simulation and measurement," *IEEE Computational Intelligence Magazine*, vol. 2, no. 2, pp. 37–51, 2007.

[92] Q. Lu, Q.-L. Han, X. Xie, and S. Liu, "A finite-time motion control strategy for odor source localization," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 10, pp. 5419–5430, 2014.

[93] Y. Tian and A. Zhang, "Simulation environment and guidance system for auv tracing chemical plume in 3-dimensions," in *2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)*, vol. 1, pp. 407–411, IEEE, 2010.

[94] Q. Lu, Q.-L. Han, and S. Liu, "A cooperative control framework for a collective decision on movement behaviors of particles," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 6, pp. 859–873, 2016.

[95] J.-y. Zhou, J.-g. Li, and S.-g. Cui, "A bionic plume tracing method with a mobile robot in outdoor time-varying airflow environment," in *2015 IEEE International Conference on Information and Automation*, pp. 2351–2355, IEEE, 2015.

[96] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[97] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[98] O. Sigaud and O. Buffet, *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013.

[99] F. Rahbar, A. Marjovi, and A. Martinoli, "An algorithm for odor source localization based on source term estimation," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 973–979, IEEE, 2019.

[100] M. L. Littman, "A tutorial on partially observable markov decision processes," *Journal of Mathematical Psychology*, vol. 53, no. 3, pp. 119–125, 2009.

[101] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[102] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[103] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[104] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 117, pp. 11–28, 2016.

[105] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and structural biotechnology journal*, vol. 13, pp. 8–17, 2015.

[106] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[107] J.-S. Jang, "Anfis: adaptive-network-based fuzzy inference system," *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[108] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, "Backpropagation: The basic theory," *Backpropagation: Theory, architectures and applications*, pp. 1–34, 1995.

[109] J.-S. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 378–406, 1995.

[110] A. Al-Hmouz, J. Shen, R. Al-Hmouz, and J. Yan, "Modeling and simulation of an adaptive neuro-fuzzy inference system (anfis) for mobile learning," *IEEE Transactions on Learning Technologies*, vol. 5, no. 3, pp. 226–237, 2011.

[111] L. Wang and S. Pang, "An implementation of the adaptive neuro-fuzzy inference system (anfis) for odor source localization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2021.

[112] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[113] H. J. Kelley, "Gradient theory of optimal flight paths," *Ars Journal*, vol. 30, no. 10, pp. 947–954, 1960.

[114] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[115] S. S. Girija, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *Software available from tensorflow. org*, vol. 39, no. 9, 2016.